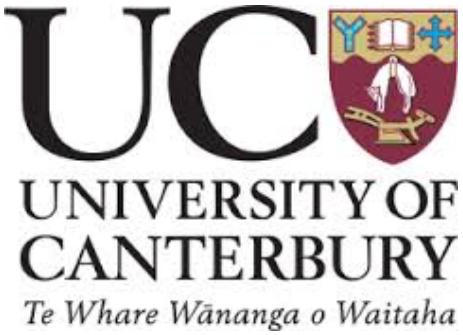
Inference of Vine Structures From Images



A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

PHD IN COMPUTER SCIENCE

BY
Ricardo David Castaneda Marin

SUPERVISORS:
Richard Green
Tom Botterill

June 2016

Abstract

Making inference and extracting information out of images is one of the most important aspects that computer vision methods are presented with. From the many techniques that exist, Markov Random Fields has been found to be a powerful mathematical tool that is both flexible – it can be adapted easily to different applications; and that has the ability to model, with different depths of complexity, the uncertainty that is found in many vision problems.

In this PhD I propose and evaluate comparatively a novel Hidden Markov Model for modeling and extracting *vine structure* from images, that is, hierarchical connectivity of vine canes. Extracting canes from vine images is a challenging problem given there are many occluded regions and overlapping canes present in such images. Previous research in the area of modeling trees and plants in images make use of manual intervention for solving the mentioned issues, or they make assumptions of the input images that are not valid in my setup. My proposed model aims to tackle directly the inference of connectivity between visible parts of canes, it is fully automatic and can be adapted to different structures other than vines. Here, connectivity inference can be done using any MAP inference method. Therefore, I have selected four methods for comparison, which are Iterated Conditional Model, Simulated Annealing, a heuristic random search based on Gibbs Sampling and Belief Propagation. These four methods are commonly used in computer vision for solving similar tasks to vine structure retrieval. In this thesis, I show comparative results of my proposed methods against manually annotated ground truth data. My Markov model and MAP

inference methods generalize and achieve two times higher precision values when compared to prior research. They also compare similarly to heuristic methods for vine structure that were developed as a part of the same project this thesis belongs to. Furthermore, I also analyzed experimentally the convergence of the selected inference methods using vine images from which I know the true optimum value and conclude that Gibbs sampling achieves better performance in comparison to the other methods that usually get stuck at local optimums.

Finally, the architecture of the system proposed in this thesis is similar to current methods in image parsing and scene understanding in computer vision. The results indicate that my proposed Markov Model together with the selected Maximum A-Posteriori–MAP inference framework are state-of-the-art methods in computer vision applied to the problem of vine structure extraction.

Contents

List of Figures

List of Tables

Publications

The following is a list of peer-reviewed papers that I have written along the course of the research presented here in this document.

1. Marin, R.D.C.; Botterill, T.; Green, R.D., "*Split-and-merge EM for vine image segmentation*," in Image and Vision Computing New Zealand (IVCNZ), 2013 28th International Conference of , vol., no., pp.270-275, 27-29 Nov. 2013.

   Summary: In this paper I propose to use Mixture of Gaussians for defining cane segments. The main idea to use information criteria from model selection theory to guide directly the split-and-merge steps for learning Mixture of Gaussians via Expectation Maximization. The method suffered from not being consistent in defining cane segments, since many of them would have holes, or intersections.

2. Marin, R.D.C.; Botterill, T.; Green, R.D., "*Gibbs sampling for 2D cane structure extraction from images*," in Automation, Robotics and Applications, 2015 (ICARA) 6th International Conference on , vol., no., pp.461-465, 17-19 Feb. 2015.

   Summary: In this paper I propose a bottom-up approach to extract vine structure by firstly segmenting an input image into cane parts, and secondly inferring their connectivity by using Gibbs Sampling. The architecture of this system is the same one as used here in my methods. However, in the paper we used custom cane segments extracted from a binary scanning algorithm rather

than skeleton curves. The Gibbs sampling method is explained in detail in Section 4.3.3 of this thesis.

3. Marin, R.D.C.; Green, R.D., "*A Hidden Markov Model For Modeling and Extracting Vine Structure in Images*," in Image and Vision Computing New Zealand (IVCNZ), 2015 30th International Conference of. Nov. 2015

   Summary: In this paper I present the same Hidden Markov Model as in this thesis for modeling and extracting vine structure from images. Here I used skeletonisation and polylines to model cane segments and I used simulated annealing to optimize an energy function defined in terms of attributes observed for each connection. The content of this paper is presented along the methods of this thesis, however, the paper is limited to simulated annealing, and it does not take into account connections flow / directions as I do here in Chapter 3.

Acknowledgements

First of all, I would like to thank my supervisors Richard Green and Tom Botterill. Their feedback, advice, technical guidance and support until the very lasts months were the best I could imagine for surviving the ups and downs of writing a PhD thesis.

Secondly, I would like to thank my parents and all my family for helping me to get up to this point in my studies, and for their support and faith in my academic career.

Thirdly, I would like to thank my wife Elaine Vivian for her love, patience and support while writing this thesis.

Finally, I would like to thank all people that I meet here in New Zealand, friends that shared with me either research or non-academic activities, and who contributed to a balanced and healthy PhD life.

1

Introduction

1.1 Context of the Problem

The research presented in this thesis is part of a robot system that aims to automatically prune grape vines. The robot is composed of a vision system and a robot arm installed inside a vehicle straddling vines. An illustration of this system is illustrated in Figure 1.1. The current pipeline of the vision system is depicted in Fig-

Figure 1.1: Robot for automatically pruning grape vines.

ure 1.2. Firstly, a set of LEDs and three cameras illuminate and capture stereo frames of the vines. The vehicle is designed to block light from outside, so all images captured are under similar illumination constraints. Secondly, the system extracts 2D cane structure for each frame image. For example, for three cameras, at any given specific time the system will output three images of the same vine from different perspectives, and the system extracts 2D canes for each of these three images, yielding three 2D structures. The 2D structure information includes the location, orientation and thickness of each of the canes present in a vine image. Subsequently, the system performs stereo correspondence and matching of the three cameras extracted 2D structures, and builds a 3D model of the imaged vine. With this 3D model, an AI component decides which canes to cut, and path planning algorithms are used to move the robot arm while avoiding collisions and pruning the vine.

1.2 Problem Description

In the context of the robot system described in the previous section, in this thesis I am interested in methods for automatically extracting 2D vine information from the imaged vines. This will be referred as the *vine structure*, and includes the following data (see Figure 1.3):

2D Vine Structure

1. *Canes Graph*: Graph describing connectivity of all canes segments present in the image.
2. *Canes orientation*: Direction of every cane segment.
3. *Canes width*: Width of every cane segment.

The vine structure will be used to reconstruct a 3D model of the vine using stereo correspondence. Methods of the robot system in Figure 1.2 that are not aimed to recover the 2D vine structure of vines (e.g. 3D stereo correspondence for building a 3D model of the vine), are out of the scope of my research. Also, methods for the

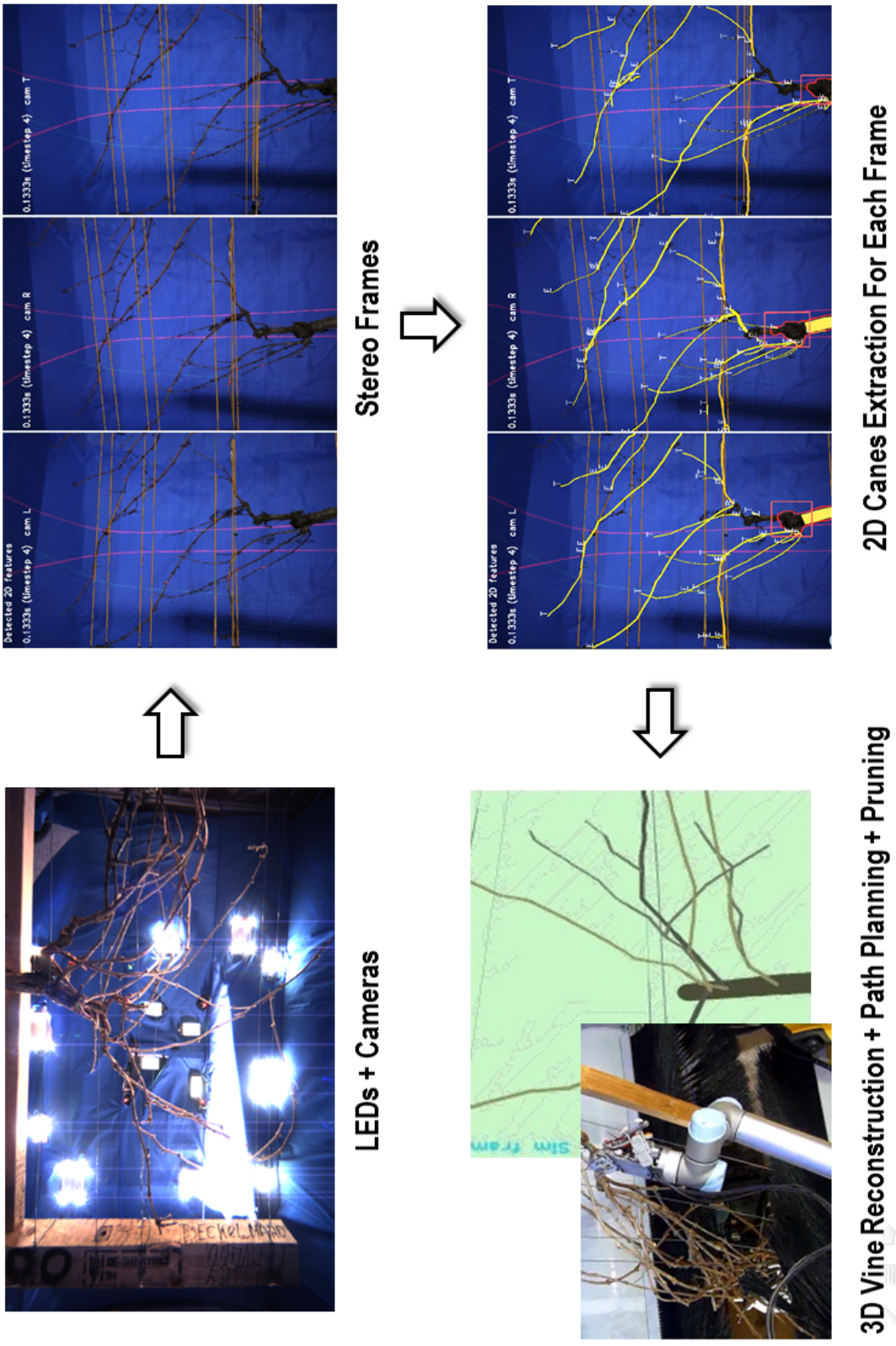


Figure 1.2: Vision system pipeline of the robot for automatically pruning grape vines.

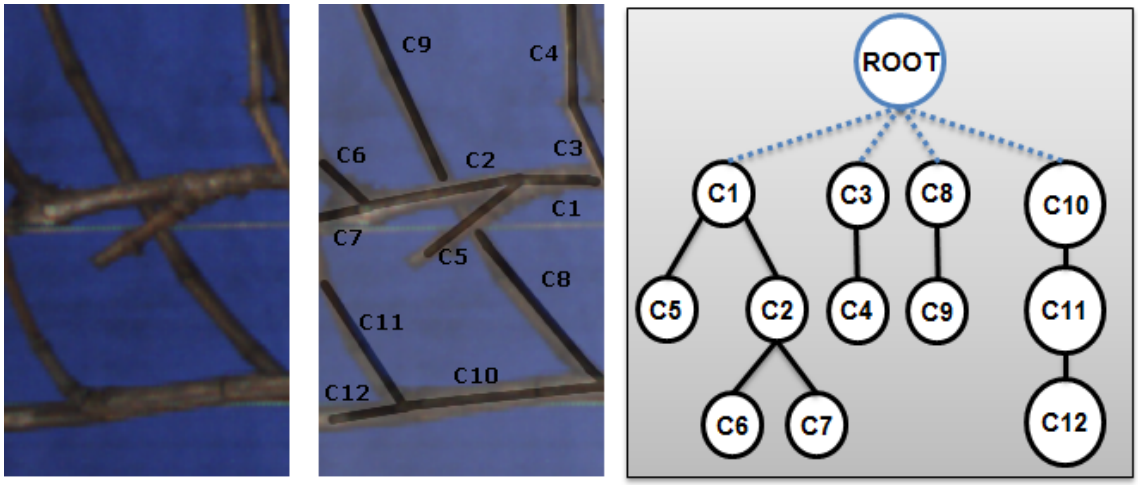


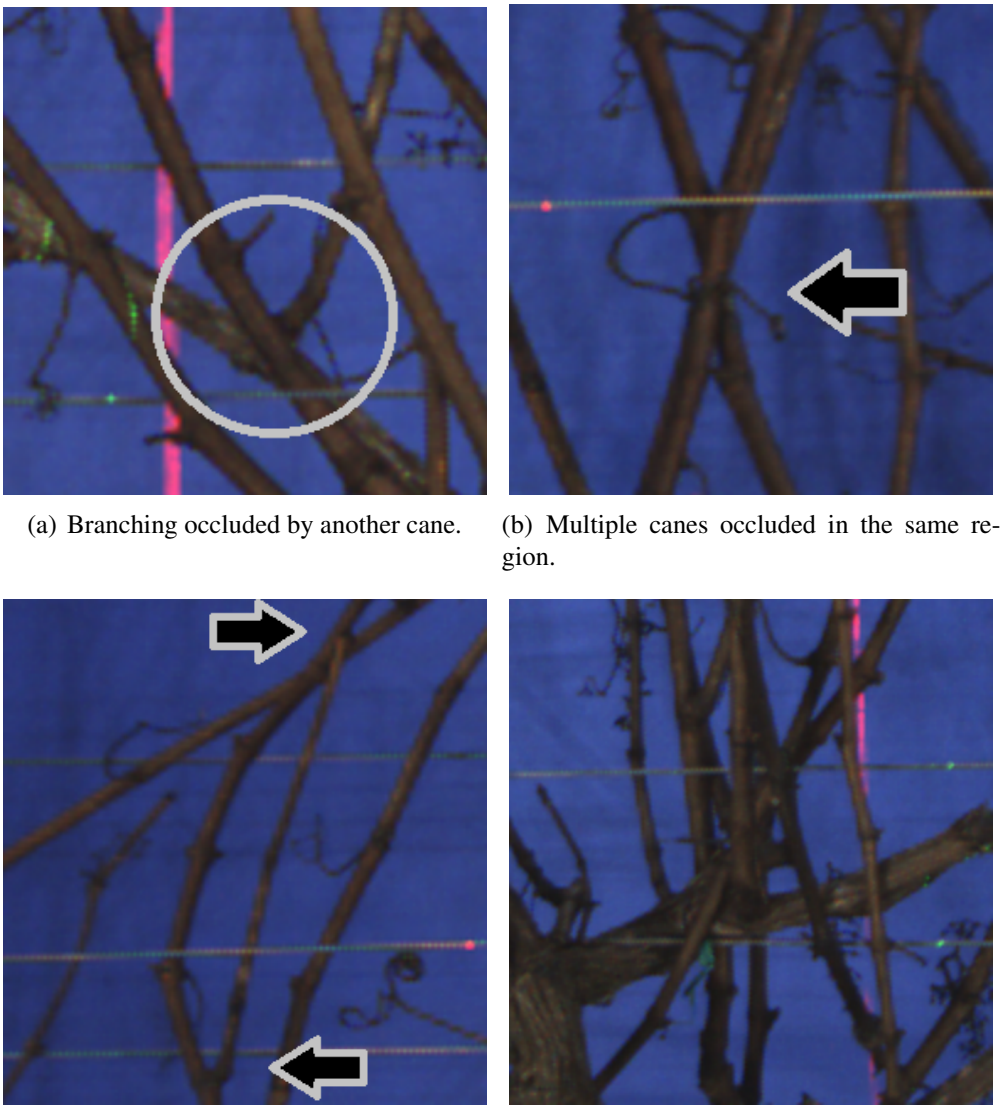
Figure 1.3: Vine structure from 2D images. Vines (left) have a tree structure (middle) which can be represented in a graph (right).

extraction of other 2D elements present in vine images such as posts, wires and other scene elements different from vine canes are out of the scope of this research.

Estimating the vine structure is a complex task. Figure 1.4 shows some interactions between canes that make estimating the vine hierarchy challenging. As it can be seen, the main source of ambiguity in cane connectivity is that of cane occlusions. Canes may occlude branching points, cane tips, and/or multiple other canes. These are not the only issues one can encounter while extracting cane hierarchy. In some cases overlapping canes have the same color making it challenging for tracking their edges and extract width. Also, canes can occur with segments perpendicular to the image plane, which makes it challenging for extracting correctly their orientation. In summary this thesis proposes and evaluates a vision model to resolve the aforementioned issues and extract the vine structure from images.

1.3 Overview of the Thesis

Given the occlusion and overlapping problems described in the previous section, recovering directly full canes from images is highly complex. This means that extracting only parts of the canes that are not occluded and that do not overlap is in



(a) Branching occluded by another cane.

(b) Multiple canes occluded in the same region.

(c) Cane tips finishing at non background regions.

(d) Combinations of (a) and (b)

Figure 1.4: Different cane occlusions and connectivity issues.

general easier. Furthermore, occlusions can be modeled and solved by performing inference of connectivity between the observable cane parts.

In this way, this PhD thesis proposes and evaluates a probabilistic model that is suitable for modeling cane segments and connectivity between them. My approach is divided into two major subsystems; *Cane Segments Extraction* and *Cane Connectivity Inference*. See Figure 1.3 as a reference. In the first subsystem, the goal is to subdivide the image into several cane parts or segments that are directly observable from images. These segments are named in Figure 1.3 from $C1$ to $C12$. In the second subsystem, a vine model is proposed where state-of-the-art inference methods in computer vision can be applied to build the vine structure graph. This thesis describes in detail the definition of this probabilistic vine model, and also evaluate four commonly used inference methods for recovering vine structure from images.

The content of this thesis is structured as follows:

- **Chapter 2**: This chapter looks into related research containing modeling and extracting tree shapes and hierarchy/structure of trees and plants. This includes skeletonization, part-based modeling, and bottom-up/top-down methods in computer vision. I will also analyze which method is better suited for the vine structure inference problem.

- **Chapter 3**: This chapter proposes a novel vine structure model using a Hidden Markov Random Field. The chapter starts by describing in detail the subsystem of *Cane Segments Extraction* that uses skeletonisation on binary images to model observable cane parts. Then, it describes in detail the construction of the proposed Markov model, and formulates everything in a way that any state-of-the-art method for estimating the Maximum A-Posteriori can be applied to extract cane connectivity and vine structure.

- **Chapter 4 - Chapter 5**: In Chapter 4, I select four state-of-the-art inference methods for using with the proposed vine model. Each method is described in detail including the algorithm being used. Then in Chapter 5, I present results of the inference methods. All methods are compared to each other with respect

to the structures retrieved, accuracy and recall against ground truth and prior research. Evaluation methods are described as well in detail.

- **Chapter 6**: This chapter summarizes my findings, gives directions for future research, and concludes this document.

1.4 Main Goal and Contributions

The main goal of this thesis is to propose and evaluate state-of-the-art methods of probabilistic inference in computer vision applied to the problem of vine structure extraction as described in the previous subsections. With this goal in mind, my main contributions are:

- A novel Hidden Markov Model for modeling vine structure in images. My model makes use of skeletonisation and polylines to model cane segments, both being highly used in related research that aims to model plants and trees (see Chapter 2). This implies that my model is very flexible and could be easily adapted to represent other structures different from vines.

- A novel formulation of the vine structure extraction problem as the solution to a MAP inference problem, which is a state-of-the-art framework in computer vision. In particular, I propose a novel energy cost model in terms of cane flow and connectivity; and I selected and evaluated four widely used inference methods for solving vine structure via MAP inference and using the proposed Markov model. These methods are Iterated Conditional Modes, Simulated Annealing, a heuristic random search based on Gibbs Sampling and Belief Propagation.

- An evaluation and comparison of the proposed vine structure inference methods. I compare values of precision and recall of the chosen inference methods against manually annotated ground truth data. Here I present improvements upon prior research by achieving more than twice higher precision values, while keeping similar recall values.

2

Related Research

In this chapter I describe and discuss related methods for estimation of vine structures in images. In the methods that I propose for modeling vines and inferring structure in following chapters, there are two fundamental branches of topics to be reviewed. The first area of research I must review is that of methods for modeling and representing plants and trees in images. Here I will observe that, similar to my proposed method, skeletonisation of binary images is a commonly used approach to represent and model tree structures and hierarchy of articulated objects including plants. Other popular methods include models for how plants grow and representations for tree branches based on geometric primitives like cylinders. A literature review of this area of research is presented in Section 2.1.

The second area of research that I must review is that of finding structure in images. Finding hierarchical structure of objects in images, or in general, modeling and extracting relations between objects in a depicted scene is a fundamental and broad problem in computer vision. It includes topics about scene understanding, bottom-up/top-down image parsing/grammars and part-based modeling in computer vision. Many of these methods, including my proposed approach, are formulated in the context of Maximum A-Posteriori – MAP estimation and thus are of interest for my research. Therefore, in Section 2.2, I will review probabilistic approaches to structure in images and review methods of bottom-up/top-down parsing, part-based

models and image parsing in general that make use of MAP estimation for solving structure from images.

2.1 Plant Modeling, Reconstruction and Synthesis

A recent summary of tree modeling research can be found in Lopez et al. [53] and Zhang et al. [102]. As noted by the authors, tree modeling research can be divided into two main branches. Firstly *synthesis methods*, focus on defining mathematical models for synthesizing realistic trees. The idea is to procedurally generate a type of tree or plant and simulate how it grows and evolves over time. This have been done using either shape grammars, botanic rules knowledge, physical simulations or heuristics [84, 77, 76]. Secondly *reconstruction methods*, focus on reconstructing 3D tree geometry of the plants and trees that have been sampled from the real world [89, 51, 42]. It is worth noting that some synthesis methods like the one in Kang and Quan [43] incorporate reconstruction techniques in order to learn and use information from real samples and improve the synthetic results.

Note that my problem of extracting vine structure from images is more related to reconstruction methods. This is because instead of simulating and constructing a model for generating vines, I aim to use vine image samples to extract their specific structure. The goal is that the same structure extraction procedure can be used for any input vine image. Thus, my problem is closely related to image-based plant modeling and reconstruction [71, 102], where the input is a set of images and the output depending on the application in question solves for specific information of the tree/plant, which in my case is the vine structure.

Figure 2.1 shows a general diagram that encapsulates state-of-the-art approaches to reconstruction and synthesis of plant and/or tree structures. Transition arrows in this diagram correspond to different computer vision and image processing methods used by different authors. In the literature, there appear to be three main categories for devices used for sampling plant and trees from the real world. These are scanners, time-of-flight depth cameras, and standard RGB cameras. This means that in general the input of a reconstruction system can be either 2D images, or 3D data like point clouds or incomplete 3D approximations of the target plant. There could be hybrid

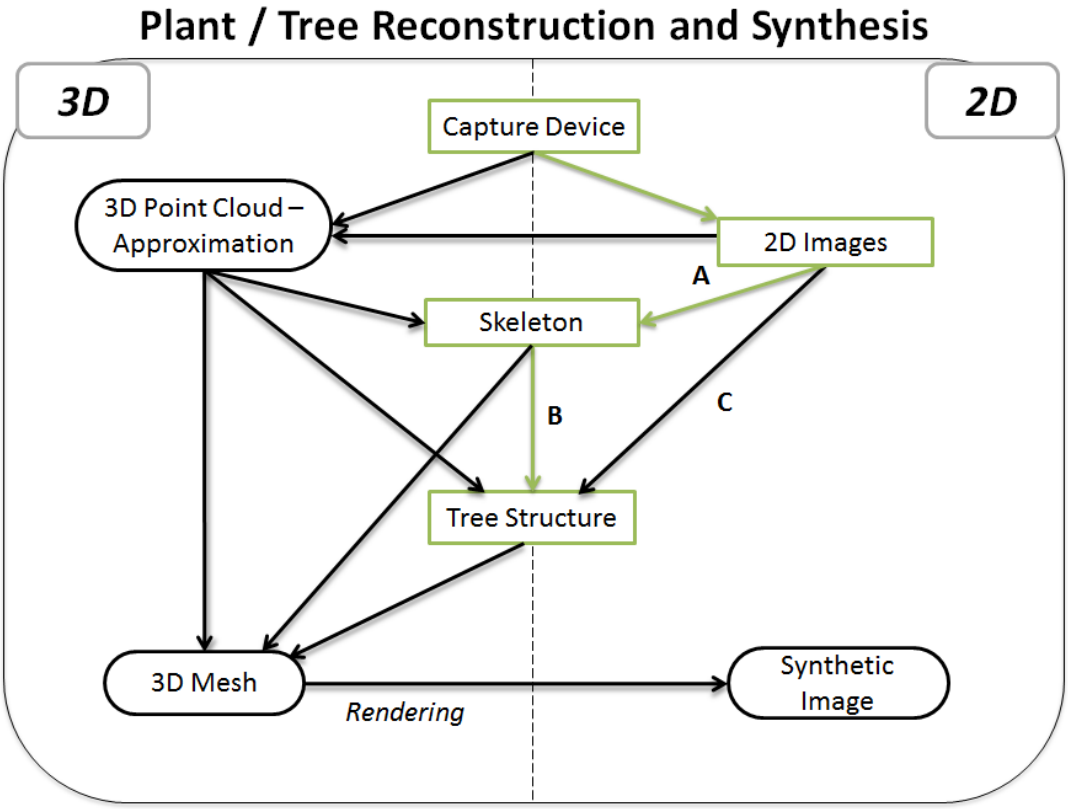


Figure 2.1: Plant/Tree Modeling, Reconstruction and Synthesis Methods. This diagram encapsulate previous research methods of modeling, reconstruction and synthesis of plants as a particular **path**. A path starts always at the *Capture Device* node, and from there, depending on the approach, it flows into the other nodes. For example, the arrow labeled A represents the different computer vision and image processing methods that have been used for getting an skeleton representation from input 2D images of plants. The diagram is also separated in the middle to differentiate methods that use 3D and 2D information respectively. My proposed method, presented in the following chapters, is shown here in green. The letters represent related methods to my approach, and they are reviewed in this section.

methods like the one presented in Quan et al. [71] which start with 2D images and then produce a 3D point cloud using structure from motion. Furthermore, recently different hybrid methods have been proposed. For example Paproki et al. [65], Kumar et al. [45] and Zheng et al. [105] used voxel carving for modeling cotton plants and plant roots respectively. Finally, Ni and Burks [62] and Dey et al. [16] used dense stereo to reconstruct models of vines and citrus trees. Differently to these methods, our pruning system follows a feature-based approach. The system uses RGB cameras that produce 2D vine images which I use as input. From here, my goal is to extract two dimensional vine structures from 2D images and skeletons so that we can use stereo correspondence/matching to reconstruct a 3D model of the vine. In the diagram of Figure 2.1 this proposed method for vine structure inference follows the path $(A \rightarrow B)$ with nodes and transitions shown in green. Thus, my methods are entirely developed in the 2D domain, and the 3D reconstruction of vines, though is part of the whole pruning system, is out of the scope this thesis.

In the following I concentrate my literature review on the methods referenced in the diagram by the letters A and B, which relates respectively to skeletonisation of tree and plant images, and tree structure computation from skeletons. Also, I include the category of methods referenced by the letter C, since they construct tree structures from 2D images without using skeletonisation, and thus they are relevant to my research.

2.1.1 Tree / Plant Structure Modeling and Fitting

In the previous section I divided plant structure research into synthesis and reconstruction methods. In this section I review methods of modeling and fitting plant models with the aim of both synthesizing and reconstructing plant structures, and in relation to my problem of modeling and extracting vine structure from images. In summary, in modeling methods one is interested in describing formally a plant in a structured manner from the topological, geometrical, physical and chemical properties which may be involved in its growing and its interaction with the environment. On the other hand in fitting methods, one is interested in taking a given model and represent real world plants through the model. For example, in a parame-

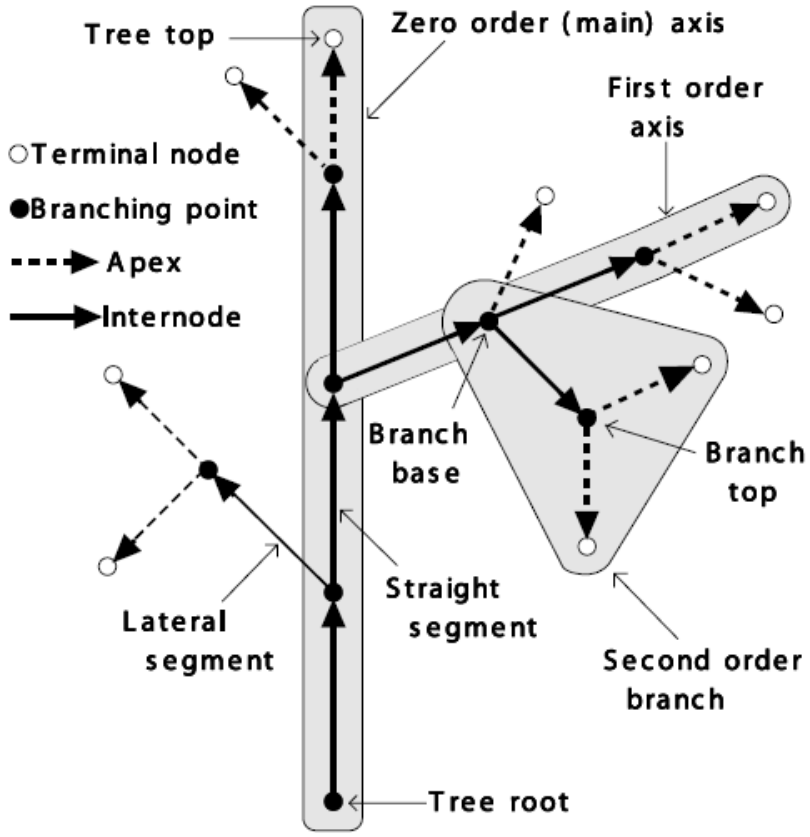


Figure 2.2: An Axial Tree Structure to describe the topology of plants. Adapted from Prusinkiewicz and Lindenmayer [69].

terized model, fitting would refer to computing the parameters of the model given an observed plant. These two topics are reviewed with detail in the following.

Plant Structure Modeling

Modeling of plant structures in its simplest form can be done by using hierarchies of parts, that is, branching structures and tree-graphs [55, 40]. Here for simplicity one also assumes that the plant scale is fixed, which means the model is restricted to a snapshot of the plant during its life-time. In Prusinkiewicz and Lindenmayer [69], *Axial Trees* are defined to complement simple tree-graph structures with the botanical notion of branches. Figure 2.2 illustrates this concept. An axial tree is often seen as a purely topological description of the plant, since no information specific to a species

or interaction with the environment is taking into account in this model. Similar to my proposed model, each branch in an axial tree is decomposed into segments, which are inter-connected to each other by node points in a hierarchical order according to a main branch. Axial trees can be augmented with rewriting rules and/or *L-systems*. The augmented models are mainly used to produce synthetic results [69, 40], but they are hard to use for modeling real world plants because a mechanism for constructing the set of rules from real samples of a determined species is unknown or cannot be easily defined in the general case [69].

A more general and robust plant model is that of Functional-Structural Plant Model-FSPM defined by Godin and Sinoquet [32]. A FSPM is able to simulate the evolution over time of the plant structure while taking into account the plant physiology [64]. The name comes from the fact that the model is a mixture of a *structural* component with a *functional* component. Here structure refers to the network of spatially interconnected plant primitives, that is, a topological description of the plant; and the functional component refers to the physic-chemical processes involved in the growth of the plant (e.g, photosynthesis, water transportation, etc.) [96]. Recent surveys on FSPM can be found in Sievanen et al. [84] and Hanan et al. [39]. Early works in FSPM were done by Godin et al. [31] when researching multi scale representations of plant structures. This model proposed an attributed graph to describe the plant structure at multiple scales. This is illustrated in Figure 2.3. In this model's structure graph, connections between nodes are associated with a function that can take as values any of the symbols $+$ and $<$. Here, the $+$ symbol indicates that a *growing node* has branched into other plant parts; and the $<$ symbol is used merely as an ordering of successive *internodes*—plant parts with a succession link to another plant parts that do not present any branching. In contrast to my proposed model, a multi-scale plant model uses attributes of angles between connection of plant parts and length of each of node. However, my model is simpler in the sense that I do not differentiate between inter-nodes and growing units. Furthermore, in my model the graph nodes are connections and there is no need to differentiate between links between plant parts, because the plant topology is only implicitly described (see Chapter 3).

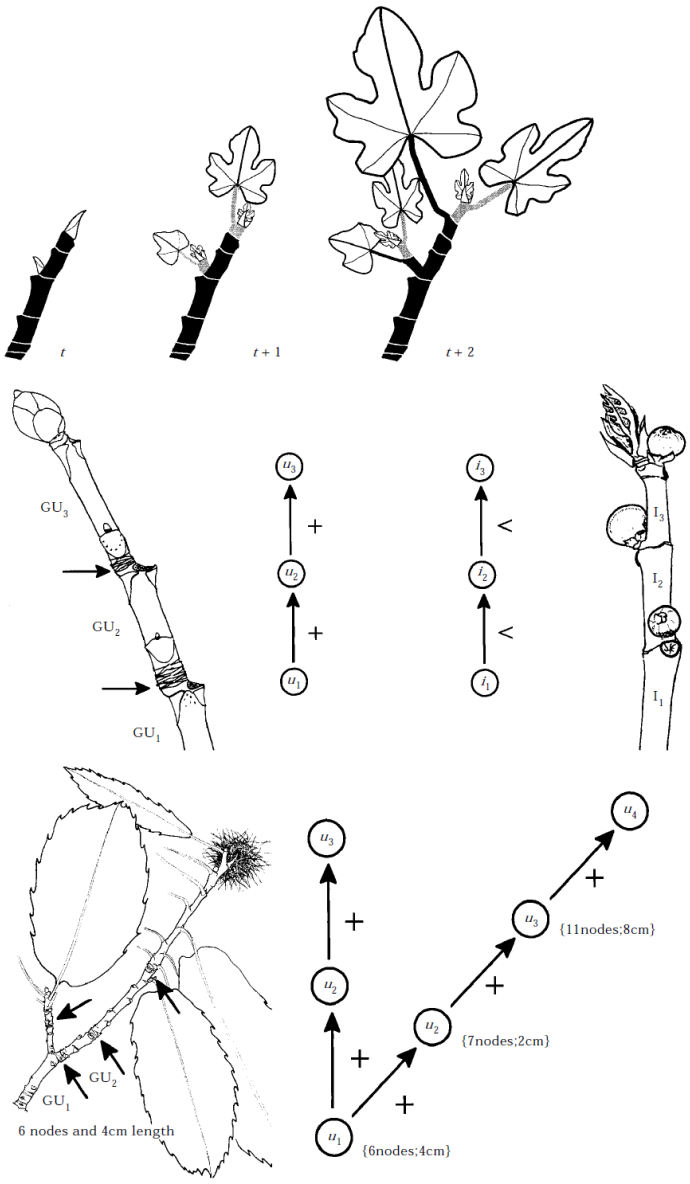


Figure 2.3: Multi-Scale Plant Structure Model. In the top, the evolution over time of a plant topology. Here, the topology of the structure at time t shown in black is not affected by the growing mechanism which produce new branches shown in gray. In the middle, the two different types of connections between plant parts. Growing units are represented with the letter u and internodes are represented with the letter I. In the bottom an attributed graph model to describe the structure of the plant. Adapted from Godin et al. [31].

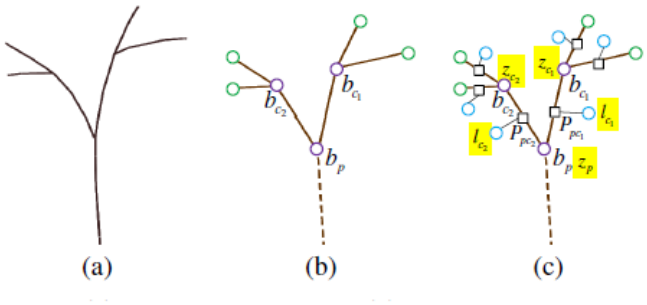


Figure 2.4: Markov Random Field for the modeling a tree structure in Chen and Neubert [12]. In (a) the input user sketch. In (b) the Markov tree for the input; and in (c) the factor graph of (b) for performing inference. Adapted from Chen and Neubert [12].

Axial trees and the early model of Godin et al. [31] only incorporated topological and temporal rewriting relations between nodes. In Ong et al. [64, 63], FSPM were coupled with graph grammars to address multi scale representations of functional properties of plants. This resulted in a complex model were the graph structure is a layered description of botanical parts of the plant at different scales. In this kind of structure descriptions, all processes involved in the growing of plants such as cellular divisions are modeled as well in a relational graph [63]. Similar plant structures as layered/multi-scale graphs have been introduced as well in [3] and has been recently used in synthesis of trees from sample images by Zhang et al. [102].

Finally, the method of Chen and Neubert [12] is the closest related to my proposed solution of modeling plant structure. This is shown in Figure 2.4. The ultimate goal of Chen and Neubert is that of synthesizing realistic trees from a set of user input strokes. In this approach, an input sketch is matched to a database of tree templates to extract the most similar predefined tree structure the user is specifying. This initial tree template is then used as a prior in a probabilistic model that is able to synthesize a realistic branching structure that resembles the input template by using MAP inference. Here, similar to my methods, tree structure is modeled using a Markov Random Field–MRF. However, in this model the Markov network is as-

sumed to be a tree and is defined in a direct mapping of graph nodes to branch points and graph edges to branch segments. Also, differently to my method, the random variable defined at nodes is the depth of the end tips of branch segments, in order for the structure of the tree to be in three dimensions. Inference over this probabilistic model is performed using Iterative Conditional Modes, a method that I also consider in my evaluation for comparison of different inference methods to find vine structure from images (see Chapters 4, 5). The main difference of my approach to that of Chen and Neubert, is that my Markov network is not necessarily a tree, since in my methods this graph is constructed from the input images and not from user input. This imply that inference in my MRF is harder and can only be approximated, in contrast to a tree graph were MAP inference can be done exactly by using a method like belief propagation [7]. Given that a tree Markov network cannot be easily defined automatically from input binary images, my method is then a novel way to model tree structure with a MRF.

In summary, plan/tree structures modeling can be either purely topological or a detailed physical description of the evolution growth of the plant. Also, all plant structure models use a graph in one way or another to represent relations of different botanical entities of the plant in question. In my methods, a vine is subdivided into cane segments, and the vine structure is also modeled as a graph encoding connectivity relations between segments. However, the input vines are considered to be always at a stage of pruning, and therefore there is no need to model evolution over time. Therefore, my approach can be regarded as a non-functional structure model of a vine, similar to axial trees and the multi-scale model of Godin et al. [31], and such that it uses a MRF model similar to Chen and Neubert [12] to infer the best tree structure for an input vine image.

Plant Structure Fitting and Skeletonisation

In the previous section I noted that structure is modeled by using graphs of relations between the botanical parts of a plant in question, and it is used to describe both physical and functional parts of the plant. In this section I am interested in reviewing state-of-the-art methods for fitting or recovering a plant structure model from 2D im-

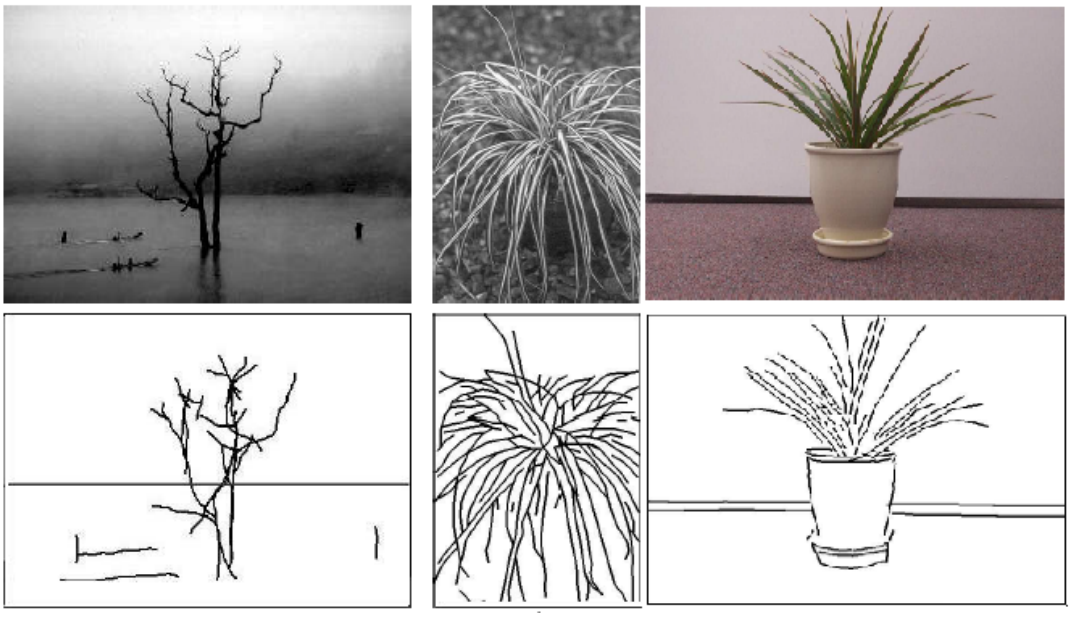


Figure 2.5: Primal Sketch for plant reconstruction. Top Input images, bottom recovered sketch. Adapted from Han et al. [36].

ages. These methods are represented in the diagram of the Figure 2.1 by the arrows that arrive to the node with the *Tree Structure* label.

As shown in the diagram of Figure 2.1, methods for recovering plant structure can be classified into methods that use skeletonisation (tagged with the letter B) and methods that build structure directly from the images (tagged with the letter C) without using skeletonisation. However, in the literature skeletonisation is used with more frequency, except for the method presented by Han et al. [36] and derived methods of image parsing and grammars [93, 107]. In Han's model, image features are used for building structure. More specifically, the author proposed to use a primal sketch [35] – a description of an image in terms of a hierarchy of features. Figure 2.5 shows some examples of this model. As pointed out by Zhu et al. [107], the primal sketch can be regarded as the first attempt to explain content in an image with context sensitive relational graphs, which is the main goal of image grammars [107]. In Han's paper, given an input image, the idea was to extract and represent the geometry and photometric structures in a graph of the plant parts in relation to other scene parts (background etc.). Inference on the graph can be used for solving occluded areas etc.

Han primal sketch method was tested on plant images, but results were found rather limited by Quan and Tan [71], because of the ad hoc probability priors set in the model.

In contrast to these methods, skeletonisation has been the method of choice for recovering plant structure from images. A recent summary and introduction of skeletonisation for plants and trees modeling can be found in Bucksch et al. [10]. Skeleton have been used in both 3D and 2D domains and in general can be computed from either 3D images of voxels, point clouds or 2D binary images. Skeletons are shape descriptors that are computed by applying morphological operations on the input set (2D or 3D). These operations can be thinning, ridge finding, medial axis transform or hybrid methods [30, 91, 53, 52]. Other forms of skeletonisation used in plant and tree modeling are principal curves [80] and the Reeb graph [10]. Despite of the method used, the aim of skeletonisation for plant or tree modeling is to represent the branches centerlines and to use this representation for computing the graph structure as seen in the previous section [10].

The two main issues with skeletonisation for finding structure of plants and trees is that of topology incompleteness and how to transform skeleton pixels into a hierarchical structure representation. Figure 2.6 illustrates these issues. Gittoes et al. [30, 29] performed a quantitative analysis of the use of skeletonisation methods for modeling vines. The authors found that, in the general case, connectivity and topology information is lost. This issue is also reflected on the literature, given that several authors device a way to tackle the recovering of the connectivity of skeletons parts and its hierarchy. For example, many methods use help from an user interface. Lopez et al. [52, 53] reported that when using skeletonisation for trees, user input refinement is often necessary for solving overlapping and occlusions between different tree segments. Also, Quan et al. [71] used manual adjustments for synthesizing trees from skeleton descriptors. In Xu et al. [99], the author built a weighted graph of scattered 3D points for approximating the skeleton by using Dijkstra's shortest path algorithm. However, in this approach the root needs to be located by the user. Tan et al. [89], developed an user interface to enable refinement of the skeleton branches. The system allowed deletion of skeleton nodes and other topological corrections for

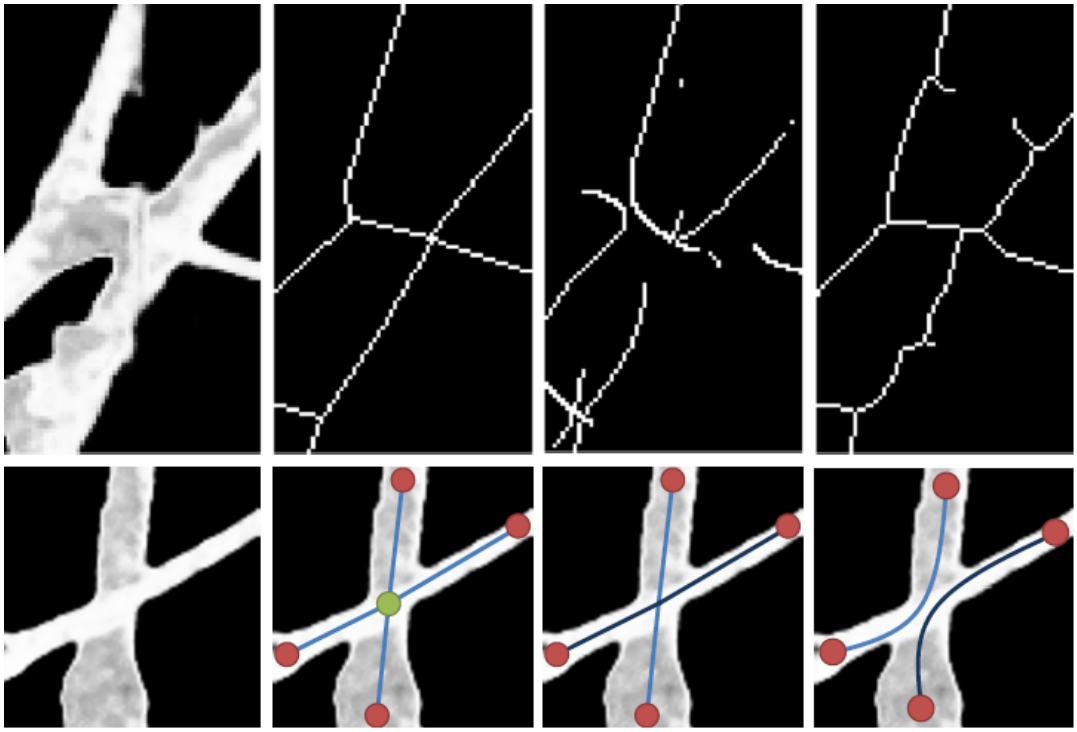


Figure 2.6: Skeletonisation for plants. In the top row, different skeletonisation algorithms to represent the shape of the input (left) vine image. On the bottom row, different interpretations that can be obtained from a skeleton representation. Images adapted from Gittoes et al. [30, 29].

improving the tree model. Also, Neubert et al. [60] reported that in his method to model realistic trees from images, better results could be achieved when using human input to place control points in the image. These controls points are used to compute a skeletal descriptor of the tree region in an input image. Subsequently, Neubert's model was extended by Tan et al. [88]. In this extended model, the author reported they used an user interface to sketch visible branches occluded by foliage of an input tree image. Also, Binney et al. [6] used user input to locate the trunk of a tree, and used a probabilistic generative procedure to fit trunk connected branches and hierarchy. Finally, Schilling et al. [79] used principal curves for recovering tree topology from TLS Data, but manual segmentation of the skeleton branches had to be performed. This segmentation is shown on Figure 2.7.

For my vine structure problem, user input is not an option, given we want the

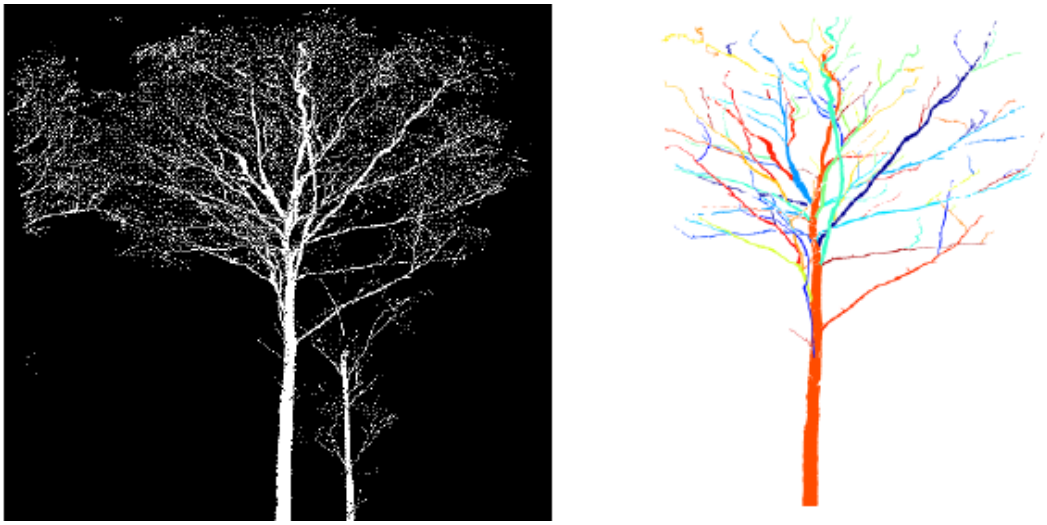


Figure 2.7: Segmentation of branches on a binary tree image. Adapted from Schilling et al. [79].

robot arm to be totally automatized to prune vines. Automatic methods for solving skeletonisation issues and hierarchical graph structure have been extensively researched as well. Livny et al. [51] defined *Branch Skeletal Graphs* for approximate skeletal structure of observed laser 3D point sets. The initial graphs are estimated assuming that different trees appear in the data as high density clusters. Then, the approach select the centroid of the cluster as the root of a tree, and minimum spanning trees are used to compute which points are more likely to be part of the same branch. At the end of this procedure, the skeletal structure is recovered, but occlusions could only be solved on small local areas. Preuksakarn et al. [67] computed skeleton hierarchy using a space colonization algorithm and principal component analysis over a set of contracted 3D points. The author reported the results were sensitive to the skeleton nodes properties they used. More recently Zhang et al. [102] reconstructed branch skeletons using a novel cylinder marching algorithm. The connectivity between the skeleton branches were solved using particle flow, and synthetic trees were built from image samples of trees. In Shenglian et al. [54] skeletons were modeled with B-Splines and they were extracted from a point cloud using Laplacian smoothing together with a filtering method of scattered points. In this method, joining of skeletons parts was performed using a priori knowledge of the plant in question,

like branching angle, leaf inclination among others descriptions referent to the tree species [54]. The authors tested the method of apple trees but found it to be non very practical given the time complexity of the reconstruction process.

Recovering automatically the structure of plants from skeletons can be challenging if no other information of the branches is taken into account. In my methods to recover structure hierarchy, skeletons are divided into different segments and attributes are given at candidates connections to help in the structure inference. Similarly, in the literature many methods augment skeletonisation representations with a measure of the width of the branches and other properties like length and mass. For instance, Lopez and Shantharaj [52] used 3D skeletons and visual hull data for reconstructing the width and real volume of plant roots. The paper reported that the input images need to have high resolution for modeling fine and small segments of the root. Similarly, Shlyakhter and Rozenoer [83] used the tree visual hull to approximate a tree medial-axis skeleton from input images. The skeleton was used as input for an L-System that solves for branch thickness and subsequently structure. Also, by using a rule based branching process, in Sakaguchi et al. [76, 77] the author modeled branch segments with not only width but also with length, mass and weight with the purpose of synthesizing realistic trees from sample images. In turn, Teng et al. [91] used skeleton thickness to discard image segmented regions that were not plausibly branches of a tree in question. In this model, the thickness at a skeleton point is heuristically computed during the skeletonisation process by counting the number of morphological operations that need to be done on the original binary image to reach the skeleton point. On the other hand, Xu and Gossett [99] researched more in depth the subject of width of tree segments. The authors made use Leonardo Da Vinci conjecture: *all branches of a tree at every stage of its height when put together are equal in thickness to the trunk*, and defined their approach taking by true that the sum of cross-sections of all children branches must be approximated equal to that of their parent. In the paper a refinement of this model is considered, where width also depends upon mass of the branch, and orientation of the children with respect to the parent is governed by a mathematical model. Similarly, Neubert et al. [60] determined thickness of branches by using Da Vinci's rule and particle flows, and

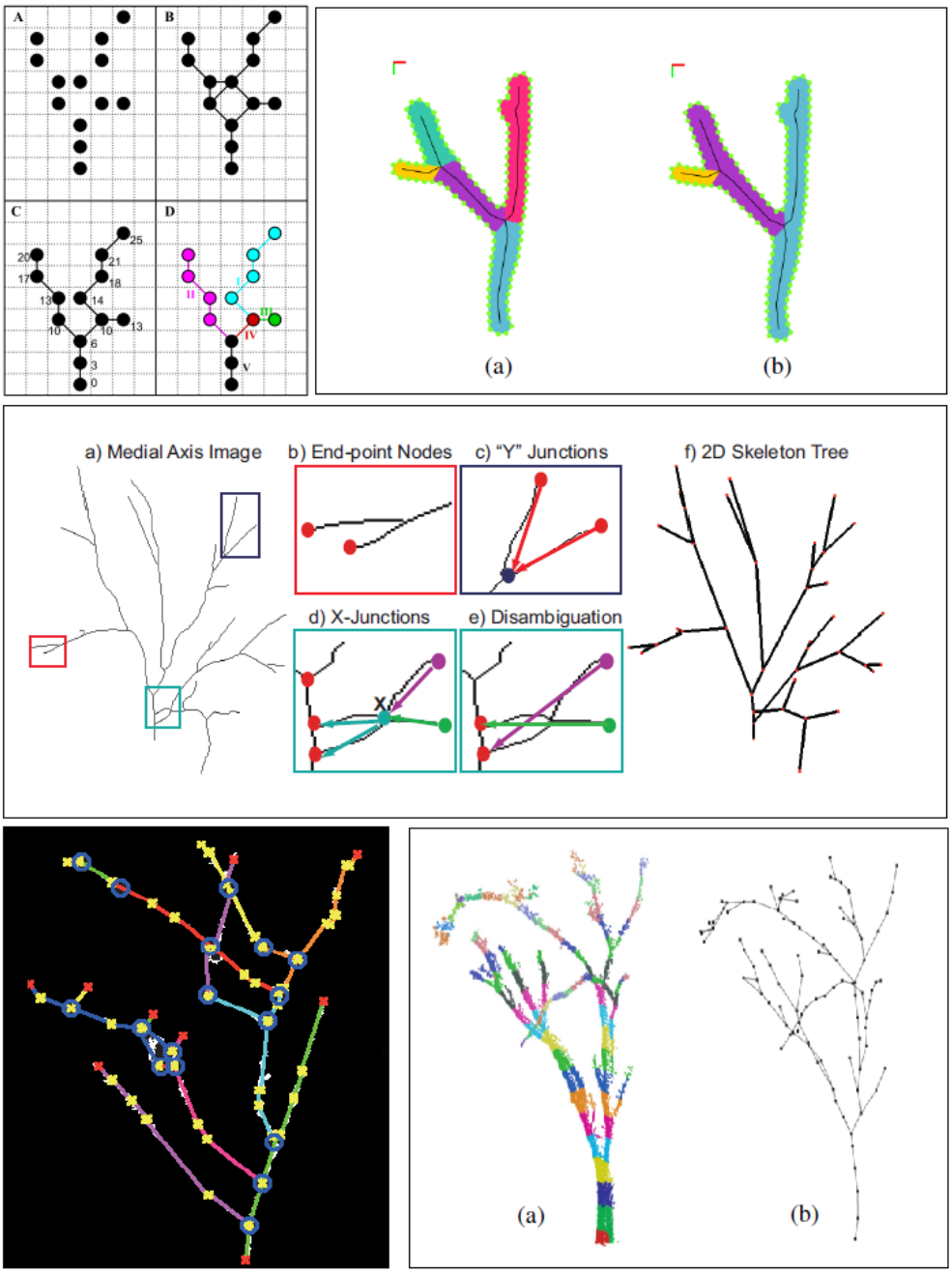


Figure 2.8: Related methods to my approach to build a hierarchy from skeletonisation. In the top row, Gorte et al. [2] to the left and Schilling et al. [80] on the right. In the middle row Lopez et al. [53]. Finally, in the bottom row, Liu et al. [49] on the left and Xu et al. [99] on the right.

Figure 2.9: Comparison of ground truth branches (right) and the branches constructed by using the approach of Gorte et al. [2, 33] (middle), from a skeleton of a vine binary image (left). Different colors represent different branches.

Tan et al. [89] used a distribution function of thickness and angle between branches to recover occlusions in between visible skeleton branches. Also, vector fields and orientation have been used for extracting hierarchy in skeletons. Guenard et al. [34] used vector fields for computing skeleton points of trunks and used a data-generated probability map for solving branches occluded by foliage. Similarly, Laga et al. [46] used a heuristic approach to trace branches from vector fields, but did not report any quantitative results of the tree structure. Finally, Cheng et al. [13] used jump edge detection, axis direction and thickness information at skeleton points (horizontal and vertical) to decompose branches into tubular skeleton segments. Here, the authors reported that solving occlusions and estimating thickness is not adequately addressed by their model [13].

Most of the methods reviewed so far make use of either 3D information, or are not suited for my vine structure problem directly given their respective limitations listed in line. More closely related to my method to build structure from skeleton are the approaches of Gorte et al. [2, 33], Lopez et al. [53], Liu et al. [49], Schilling et al. [80] and Xu et al. [99]. Figure 2.8 shows the skeleton graphs constructed and used in each of these methods. Firstly, in this figure on the top left corner, Gorte et al. [2] creates a hierarchical skeleton graph by using Dijkstra's shortest path algorithm. Given the

set of skeleton points (sub figure A) a weighted graph is constructed (sub figure B). Edges are set between adjacent skeleton pixels and weights are defined with values 3 or 4 depending if the pixels are adjacent in the main axis (X-Y) or diagonally respectively. Then a root is selected (usually the pixel with lowest y coordinate) and Dijkstra's shortest path creates a minimum spanning tree from the root (sub figure C). Branches are constructed by identifying nodes that are parents for more than one time (sub figure D). This is a heuristic method that is highly sensitive to the skeleton input and only constructs structure based on local information. As an example, Figure 2.9 shows that occlusions are not well treated by Gorte's approach. As we can see in the figure, this approach will tend to over segment branches and furthermore will construct erroneous structure on overlapping branches that are rooted in different parts of the image.

Similarly to Gorte's method, Schilling et al. [80] constructed structure graph from skeletonisation using a Depth-First search algorithm. The kind of structures retrieved there are shown in the top right corner of Figure 2.8. In this approach, the root is again the lowest y coordinate. A segmentation of skeleton points into end points– points with only one skeleton neighbor; and junctions– points with 2 or more skeleton neighbors, is performed to obtain skeletal segments (sub-figure (a)). Then, for constructing skeleton branches (sub figure (b)) only local information such as adjacency of skeleton points and curvature of the adjacent segments are taken into consideration. Therefore, this method suffers from the same problems as Gorte's approach when used on vine images for our structure inference problem (see Figure 2.9).

Next, in the middle sub figure of Figure 2.8 Lopez et al. [53] constructed a 2D skeleton graph by extending Gorte et al. [2] approach. As in Gorte's method, Lopez's algorithm start by selecting an unique root as the skeleton pixel with the lowest y coordinate. Then, it finds all end points of the skeleton and apply a shortest path algorithm to construct roads from the root to the end points. The novelty of Lopez's method in comparison to previous approaches, is that of a heuristic labeling scheme to identify two types of bifurcation/multi-furcation nodes in the skeleton graph. These two types of junction nodes are $Y-$*Junctions* and $X-$*Junctions* and are shown in Figure 2.8 in the middle sub figure with tags (c) and (d). This method also

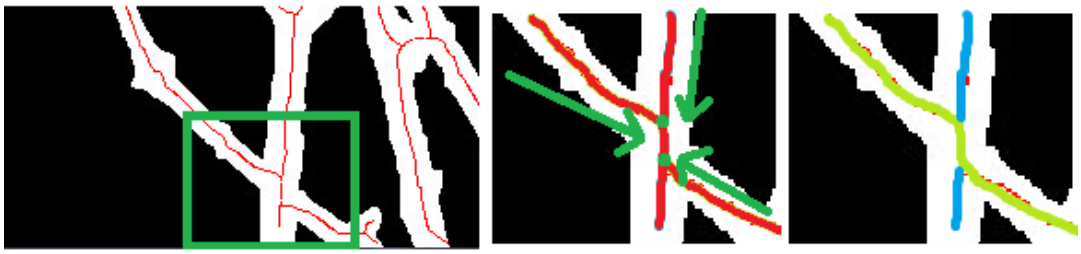


Figure 2.10: Applying Lopez et al. [53] labeling scheme to a region of the vine skeleton image on the left. In the middle, two recognized junction nodes are shown in green. These are recognized as $Y-$*Junctions* wrongly since the ground truth on the right shows this is actually an occlusion.

works only with local adjacency information on the skeleton points and the author reported that when multiple branches occlude each other by two or more branches at the same region the algorithm will fail and the structure will need to be corrected. In Lopez's method the structure was corrected with an user interface [53]. Since, multiple occlusions happens in our vine images all the time, this method cannot be incorporated directly to our vine structure problem. An example of $X-$*Junctions* that are wrongly recognized as a couple of $Y-$*Junctions* is shown for a vine skeleton in Figure 2.10.

Continuing our discussion of the methods in Figure 2.8, next, in the bottom left corner is the structure extraction method of Liu et al. [49]. In this approach, the skeleton structure is built by using the Hough line transform. However, instead of recognizing all lines at once, the author proceeded iteratively finding the line with most points and then removing those pixels from the binary set, to continue with the next iteration of line fitting. The author reported that this procedure will cut most of the junk lines that would appear otherwise when fitting lines to the entire binary set at once. The set of fitted lines however do not carry any information of structure of the tree. To address structure, Liu's method assign to each line a radius threshold that is used to match lines that are close enough. In the image of Liu's method on Figure 2.8, blue circles represent points where lines are found to be close enough. At these points, if lines also are such that the angle between them is small than a threshold, then the lines are recognized as the same branch, and thus the point is an occlusion

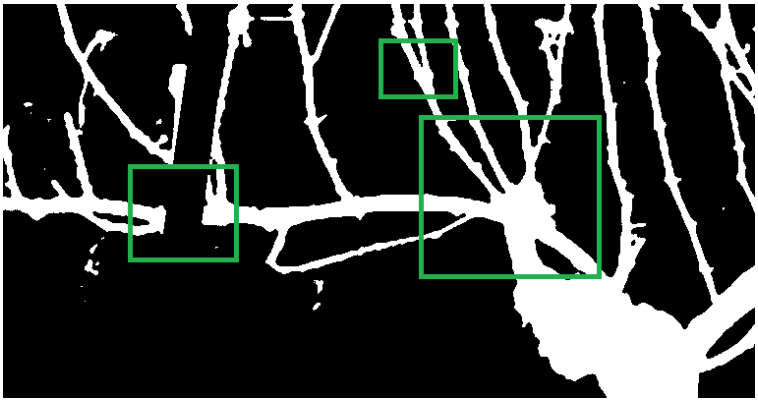


Figure 2.11: Here we have a vine binary image with a portion of a post removed, generating gaps in the map. Using the method of Liu et al. [49] on the green regions will fail to extract the correct structure. First on the left, if the threshold is too low, no connections would be made. On the right, angles at bifurcations can be small as well, and thus they can be wrongly recognized as occlusions.

point. Otherwise, the lines are recognized as different branches at a bifurcation point. In this figure, same color represents same branches, and red crosses represent their end tips. This heuristic method, as the previous reviewed methods, incorporates only local information for building structure. It also only works for simple structures. See Figure 2.11. Vine binary images present occlusion regions that are ambiguous to Liu's method, since they can mismatch bifurcation and occlusion points.

Finally, the method of Xu et al. [99] is shown in the bottom right corner of Figure 2.8. This approach start with a point cloud from which a graph of adjacency is built. Edges in this graph are assigned a weight that is proportional to the euclidean distance of the points. However, the points that are further away than a predefined threshold are not treated as neighbors and this results in a collection of disconnected subgraphs. Here, subgraphs with less than certain amount of points are discarded, to avoid connecting junk points that are possible noise from the capture device. To produce a skeleton for the tree from these subgraphs, the root of each subgraph is assumed to be known (the point with the lowest y coordinate) or is manually marked. Then, using Dijkstra's shortest path algorithm, every point in a subgraph is assigned a distance to its corresponding root. Subsequently, this distance is quantified to produce bins of points. This is shown in different colors in the image of Xu's method

in Figure 2.8 (sub figure (a)). Taking the centroid of each of these bins of points, a skeleton is built (sub figure (b)). Each disconnected subgraph is then represented by a skeleton. To built a unified skeleton, the tree root is taken and Breath-First search from this root to other subgraphs roots are computed to analyze whether the two skeleton subgraphs should be connected or not, based on distance and angle. For my vine structure problem, this method carries the same problems as Gorte et al. [2, 33] shown in Figure 2.9. Furthermore, in Xu's method the hierarchical structure happens only in between different subgraphs that possibly represent already big portions of the tree and thus different branches will be in the same node in the hierarchy. This imply that in this method is not possible to describe each branch of the tree separately, which is one of my main goals for solving vine structure.

In summary, state-of-the-art methods for solving tree structure from skeletonisation make use of algorithms for shortest paths and minimum spanning trees, and make use of local information (e.g. adjacency) and/or manual intervention. In contrast, in my methods I propose a probabilistic model where the inference of structure of vines will take into account global information together with local properties of connections between different branch segments. Being similar to Chen and Neubert [12] model, my proposed MRF method is a novel approach to automatically infer structure of trees from images. In the next subsection, I review state-of-the-art approaches for MAP estimation in computer vision which are commonly used for inference in MRF models and thus are related to my structure inference methods.

2.2 MAP and Structure Inference in Computer Vision

One of the most widely used branches of mathematics for modeling and solving computer vision problems is that of probability and statistics. In this matter, the book of Prince et al. [68] is a good source for a general view of methods for learning and performing inference in computer vision. In a probabilistic framework, a vision problem is described in terms of measurable visual data z, and an unknown state variable of interest x one wants to relate to z. A relation of special interest between x and z is the posterior probability distribution $P(x|z)$. This distribution is of interest because from it one can get the most likely state x^* that explains the measured

evidence data z, with:

$$x^* = \underset{x}{\operatorname{argmax}} P(x|z) \tag{2.1}$$

This is also known as the Maximum A-Posteriori– MAP. Vision problems that can be solved with MAP estimation are for example image segmentation where the best labeling x^* are to be solved from the observed image pixels z; solving for the disparity map x^* from captured images z; and image parsing where one is interested in the most likely relational graph x^* that describe syntactic and contextual relations between a set of image regions z.

In general, the posterior distribution $P(x|z)$ can be as simple as a Gaussian, or can be a complex non-linear and/or non-convex function. This implies that very often in practice the posterior distribution cannot be exactly optimized as in Equation 2.1 given its complexity. Furthermore, the posterior may be totally unknown. Therefore, strategies for modeling the posterior exist, and methods for MAP estimation with or without having an exact form for $P(x|z)$ have been extensively researched [68, 7].

For modeling the posterior $P(x|z)$, in the literature there are two major approaches one can follow. They are named *discriminative* and *generative* models respectively [68]. First, in a discriminative model one must specify directly $P(x|z)$ without taking into account how likely the measurements z are. In contrast, in a generative model one models the likelihood $P(z|x)$ and the prior $P(x)$ and uses Bayes's rule for the posterior:

$$P(x|z) = \frac{P(z|x)P(x)}{\int P(z|x)P(x)dx} \tag{2.2}$$

In the literature, there are named advantages and disadvantages to both discriminative and generative methods [47, 68]. In summary, each model can perform well depending on the problem in question though mixed methods can be designed as well [94, 73]. For example, it has been reported by Ng and Jordan [61] that for the task of classification, discriminative models perform better and there is no need for modeling the likelihood or incorporate prior information when the number of training samples is high enough. On the other hand, generative models can incorporate information on how the data was acquired; can incorporate expert information in the form of a prior for the unknown state x; can be extended to unsupervised learning

in an easier way; and though inference is simpler in discriminative models, generative models solutions can express dependencies of data and unknown information better [68].

Generative models are more commonly used in computer vision applications as reported by Prince et al. [68]. In particular, Hidden Markov Models – HMM are widely used for vision problems. An overview and state of the art of HMM methods in image processing and computer vision can be found in the books of Blake and Kohli [7] and Li et al. [48]. In contrast to my methods in Chapter 3, I propose to use a HMM to model vine structure, and then use Equation 2.1 to solve for the vine hierarchical structure x^* from a set of cane segments z extracted from the skeleton of a vine binary image.

For performing MAP estimation by solving Equation 2.1, many methods from energy minimization and optimization in general can be borrowed. This is because as I will show in Section 3.2 the posterior can also be modeled with an energy function that captures both likelihood and prior information, in such a way that the minimum to this energy is equivalent to the maximum of the posterior (see Equation 3.3). Therefore, methods like graph cuts, simulated annealing, dynamic programming, and other numerical algorithms for optimization can be used for MAP inference [7]. On the other hand, when the posterior is left out of an energy minimization context, Monte Carlo methods are used for representing $P(x|z)$ from a set of samples, and furthermore estimate the MAP. In Chapter 4, I review four common methods for MAP estimation in both contexts of energy minimization and Monte Carlo sampling. I deffer the reader to this chapter for details of these methods. Also, in Chapter 5 these methods are compared and evaluated for vine structure inference. Therefore, in the rest of this section I review methods in computer vision that solve for structure in images using MAP estimation and that are then related to my approach of vine structure inference. These are part-based models, bottom-up/top-down parsing and image parsing methods.

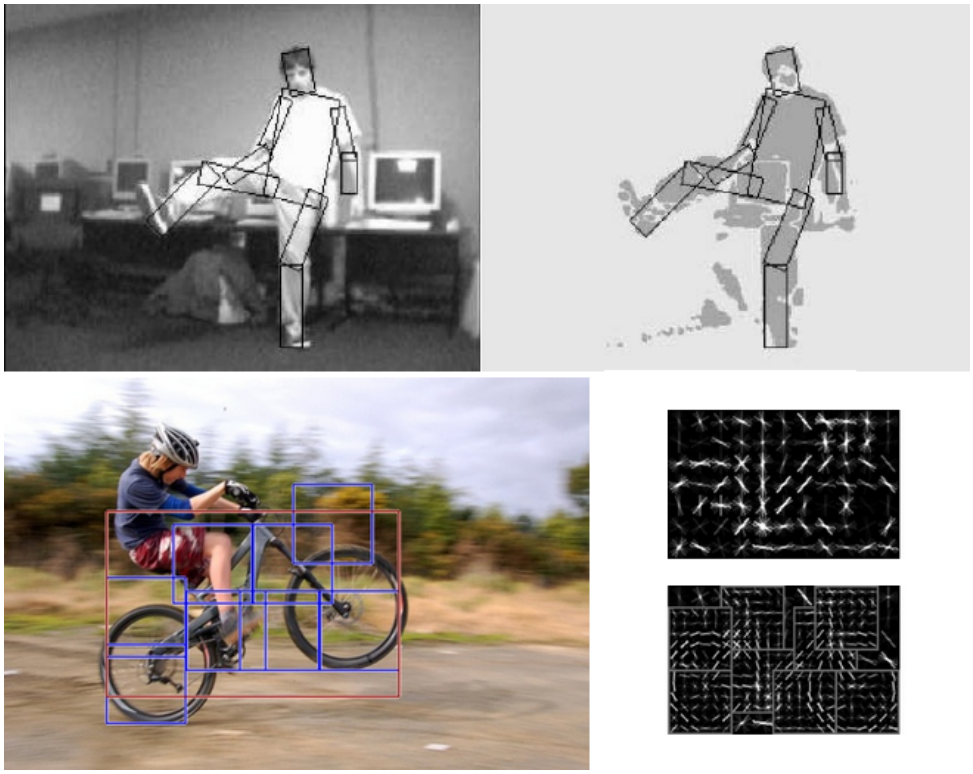


Figure 2.12: Part-based models. In the top row, a human model is composed of legs, hands, head and torso. Adapted from Felzenszwalb et al. [22]. In the bottom row, to the left a bicycle (red square) is detected from recognizing its different parts (blue squares) and analyzing the geometrical relations between these parts. To the right, the histogram of gradient features used for recognition. Adapted from Felzenszwalb and Girshick [20] and Felzenszwalb et al. [19, 21].

2.2.1 Part-Based Models

Part-based models address modeling of objects in images as a collection of fundamental parts and their geometric and appearance relationships. Examples of this kind of models are shown in Figure 2.12. Here a bicycle is composed by wheels and other body parts; and the human body is composed of legs, hands, head and torso.

Part-based models were developed by Felzenszwalb et al. [22] inspired by the early work of *pictorial structures* by Fischler and Elschlager [23]. In part-based models, relations between parts are specified by defining geometric and appearance constraints of each part relative to each other. Then an energy function on these

constraints is formulated. The energy is formulated such that fitting a part model to a new instance in an image means finding the configuration of the parts that have minimum energy (eg. the dislocation and angles between parts are in accordance to the constraints of the model).

A summary of methods developed for learning and matching part-based models can be found in Felzenszwalb et al. [22, 21]. For learning a part-based model, an unsupervised learning framework for the parameters of the model given a set of training images is defined, where each image has annotated the locations of each part of the object model. Here the unsupervised learning refers to the connections between the parts, not finding automatically the parts. Other learning techniques include standard maximum likelihood parameter estimation, computation of a minimum spanning tree for the connections between the parts, and classification methods like Support Vector Machines [22] and Linear Discriminant Analysis [27, 25]. For object detection, the matching of a part-based model to new image instances is done via MAP estimation. This is solved using a modification of the Viterbi Algorithm and/or similar Dynamic Programming methods [22]. Also, grammars can be incorporated to the model for addressing occlusions during recognition [17, 28, 26]. Finally, modeling of appearance and geometry between parts depends on the model. One common model for appearance is a Histogram Of Gradient – HOG pyramid [21] (see Figure 2.12).

In relation my vine structure inference problem, parts models cannot model the different number of branches and canes that a vine can have. This is because a part model has a fixed number of parts. One thing that can be done is to create a part model for joining two cane segments (two parts), and another part model for branching of one cane segment into other two (three parts). However, I will still need a way to connect these two part based models and with a variable number of them. Finally, similar to part-based models, my method uses soft constraints on connections between cane segments. These constraints are encoded by probability distributions on the attributes of the connections, and similar to part models, MAP estimation can be used for finding the most likely vine structure.

2.2.2 Syntactic Pattern Recognition and Grammars

Part-based models is not the only method that aims to describe an object in terms of relations between a set of constituent primitives. A much broader area of research for this matter is that of Syntactic Pattern Recognition which is based on Statistical Pattern Recognition. A good reference for both methods can be found in Burke et al. [11]. In Statistical Pattern Recognition, patterns are categorized in terms of a set of extracted features, and an underlying statistical model for the generation of the patterns. The idea is that by learning what features compose a determined pattern one can perform recognition. However, this strategy is limited, since relations between features that are intrinsic to the category of the pattern are not considered. Syntactic pattern recognition alleviate this by considering the specific way on which the constituent primitives are configured and/or combined together. Here, if one name the set of all relations/connections between sub-patterns as *the structure of the class*, then one can summarize that Syntactic P. R. is a form of Statistical P. R. where structure of the class is modeled as well [11].

In Syntactic P. R., structures are represented by formal grammars [11]. In this case, instances of the class are strings of the grammar. This gives raise to three key problems. First, one need methods for extracting the structure of a determined class. This is referred as *grammar inference* [11]. A popular method used for this is that of using machine learning techniques to learn the grammar from a set of training strings from the same class. The second problem is that of *parsing*. In parsing, given a new string sample and its class grammar, one wants to reconstruct the sequence of grammatical rules that leaded to the input string. Methods for parsing in images and computer vision are considered in the next subsection. Finally, the third problem is *classification*. Here, given a string we would like to decide to which class it belongs. This is equivalent to infer the grammar from which the string was generated [11].

Syntactic P. R. methods are mostly applied in computer vision for image recognition, classification and understanding [11]. Early methods like Fu et al. [101, 92] had been found to be limited, since primitives used to define image structure are usually hard and/or unreliable to find, and also because the grammar model is often itself limited [37]. Recent approaches have extended Syntactic P. R. methods to use

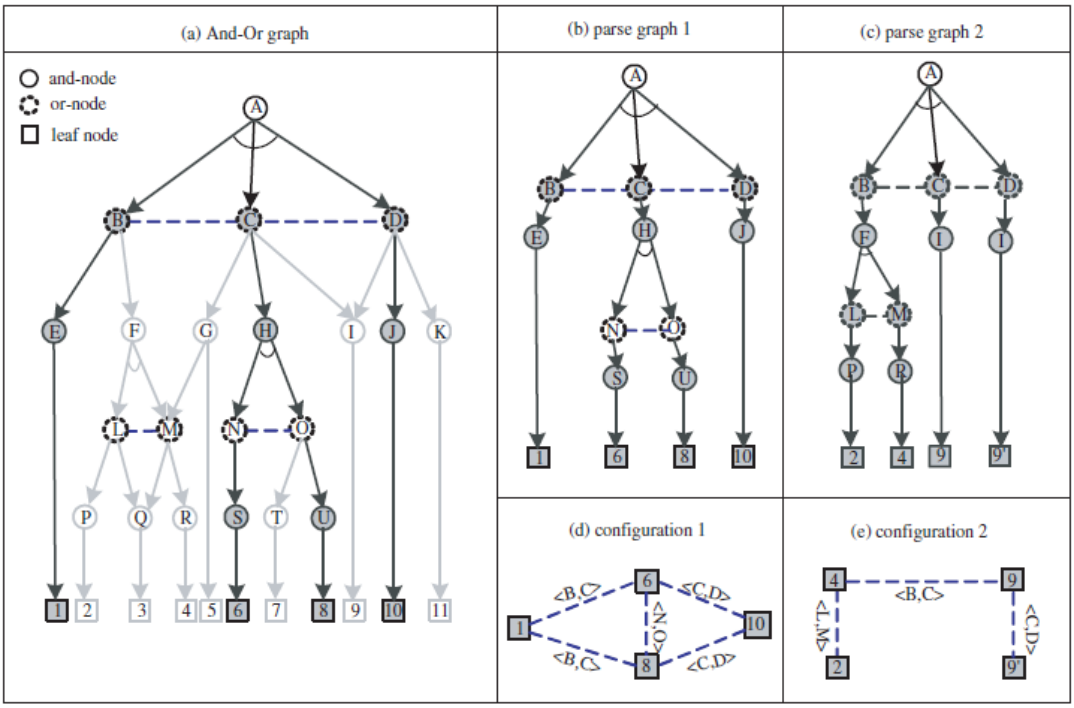


Figure 2.13: And-Or graph in image grammars. To the left the And-Or graph encode all possible configurations of the leaf nodes. To the right some examples of these configurations. Adapted from Zhu et al. [107].

stochastic context-sensitive grammars like Zhu et al. [107] and attribute grammars as in Matsuyama et al. [59] and Han and Zhu [37]. The name *image grammar* was established in the paper "An Stochastic Grammar of Images" by Zhu et al. [107]. However, early research that used grammatical rules for image parsing can be found in Stiny et al. [85] and Han and Zhu [37]. Also, some models in the literature, like Hierarchical Compositional Models – HCM by Zhu et al. [106] or Composition Systems by Geman et al. [5], can still be referenced as image grammars, given than their formulation are based on a formal grammar specification and their goal is to recognize and/or parse objects in images [1].

In summary, an image grammar is composed of an And-Or tree/graph (Figure 2.13), where parsing is done using a *bottom-up/top-down* framework (Figure 2.14).

[1]Note however that the difference between these methods is their grammar representation used. Image grammars in the literature are methods that exclusively use an And-Or tree/graph representation.

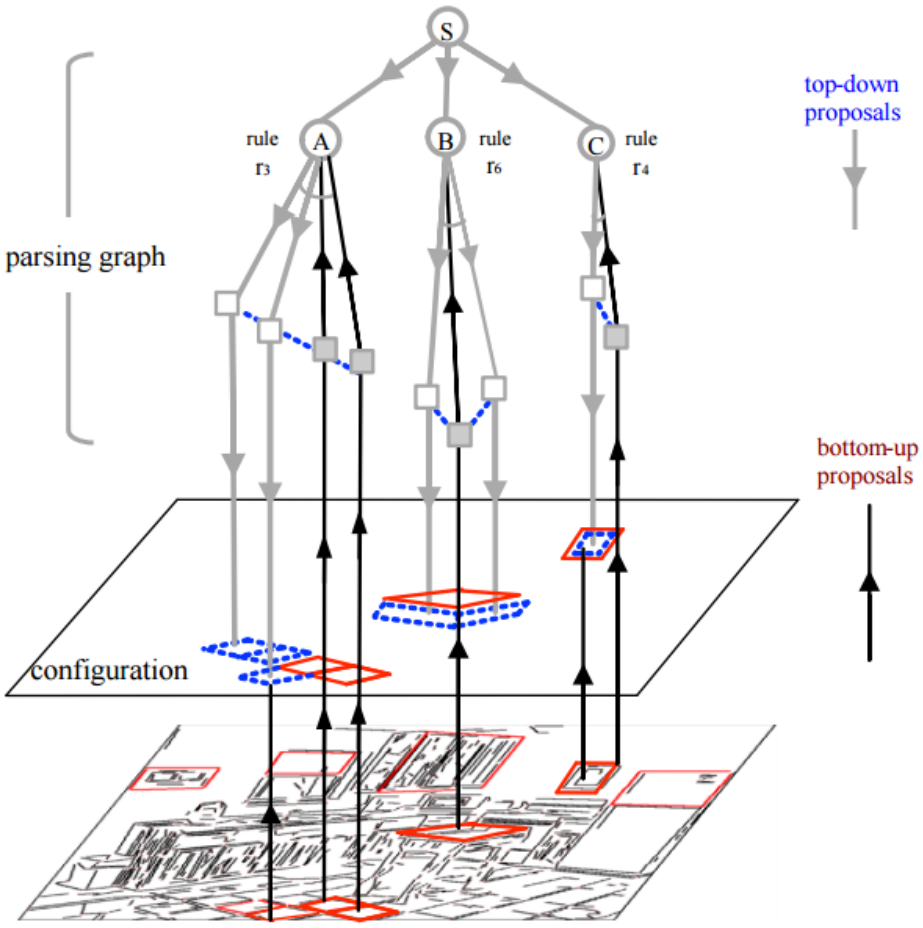


Figure 2.14: Parsing an image using a bottom-up/top-down framework. Adapted from Han and Zhu [37].

Here, the strategy for parsing an image is to use standard vision methods to find image parts that build the nodes on the And-Or graph, and then use probabilistic inference methods to recover the most likely structure (relations/configuration) of the parts [107].

Image grammars, and in particular image parsing are related to my methods of vine structure inference. In fact, parsing an image usually requires solving the optimization problem of finding the best configuration graph of the model given the image and the nodes in the graph. Most of the time, configuration spaces are huge, and brute force methods are not an option [93]. This is similar to my model, where I must solve for the most likely configuration of vine structure in a Markov network and in a high dimensional space. Predominant optimization methods that are used during parsing are reviewed in the following section, including bottom-up/top-down methods, Data-Driven Monte Carlo, and greedy optimization techniques.

2.2.3 Image Parsing

In the context of formal languages, parsing refers to the analysis and interpretation of strings in terms of the rules of their grammar. There are two types of parsing one can encounter on the literature that are related to computer vision. Firstly, *bottom-up parsing* refers to simplification of an input string as to represent it as the root symbol of the grammar. The standard approach is to use grammar rules backwards, i.e., rules whose right hand side matches the input string, parts of it, or subsequent simplifications of it, until reaching the root symbol. Therefore, parsing is done from the input string (bottom-up) to the root symbol of the grammar. Conversely, *top-down parsing* starts with the root symbol and then consider the set of rules that can be applied to it. This time, the approach is to decompose into parts the root symbol or its subsequent sub-strings by using the rules whose left hand side matches them. This decomposition is guided as to reach the input string from the root symbol (top-down). In any of the parsing methods, if more than one rule can be applied in any given step, one have to follow some criteria to select one of these rules and still break-down or glue-up the string into representative parts, e.g., in a greedy parsing algorithm one would choose the rule with the highest probability [37].

To relate parsing to images, bottom-up parsing can be intuitively thought as going from primitive parts of an image (e.g. segmented regions, group of pixels, etc.) to a representation by relations between them. This method of parsing is in the literature associated with discriminative models [93, 103], since the probabilistic model is conditioned to the image regions observed in target images.

On the other hand, top-down parsing in image grammars refers to a decomposition of the image into image regions generated from the grammar rules. In contrast to bottom-up parsing, top-down parsing is associated to generative models [93, 103]. Here one assumes to have a full probabilistic model from which one can generate all different configurations of image regions, that is, all kind of images according to the expressiveness of the model.

Bottom-up and top-down parsing, in relation to my vine structure problem, have always associated an inference methodology. Han et al. [37] parsed configurations of rectangles in images using an attributed grammar for composition of rectangles. The author reported that for his problem, a steepest ascent algorithm such as best first search could be used; however they outlined a novel heuristic iterative method of bottom-up/top-down rectangle compositions in the image. This is outlined in Figure 2.14. The optimization method was linked to find the optimal parse graph and perform MAP inference [37]. Similarly, in Tu et al. [93], the authors related bottom-up/top-down to MAP inference with the purpose to infer relations between segmented regions in an image. The inference method used was the Metropolis-Hasting Monte Carlo method (see Section 4.3.2) and they use text characters recognition as a case study. Inspired by the work of Han et al. [37] and Tu et al. [93], recent research follows the same approaches to MAP and bottom-up/top-down parsing in images. For instance, Liu et al. [50] used the same procedure as Han et al. [37] of greedy bottom-up/top-down parsing for outdoor scenes for building a 3D model with inferred semantic information; Zhao et al. [104] used bottom-up/top-down, simulated annealing and Monte Carlo methods to parse indoor objects with models for geometry and appearance; and Qi et al. [70] used bottom-up recognition and top-down proposals for parsing indoor scenes with cubes in perspective and using Monte Carlo methods similar to Tu et al. [93]. Also, many recent methods incorporate Markov

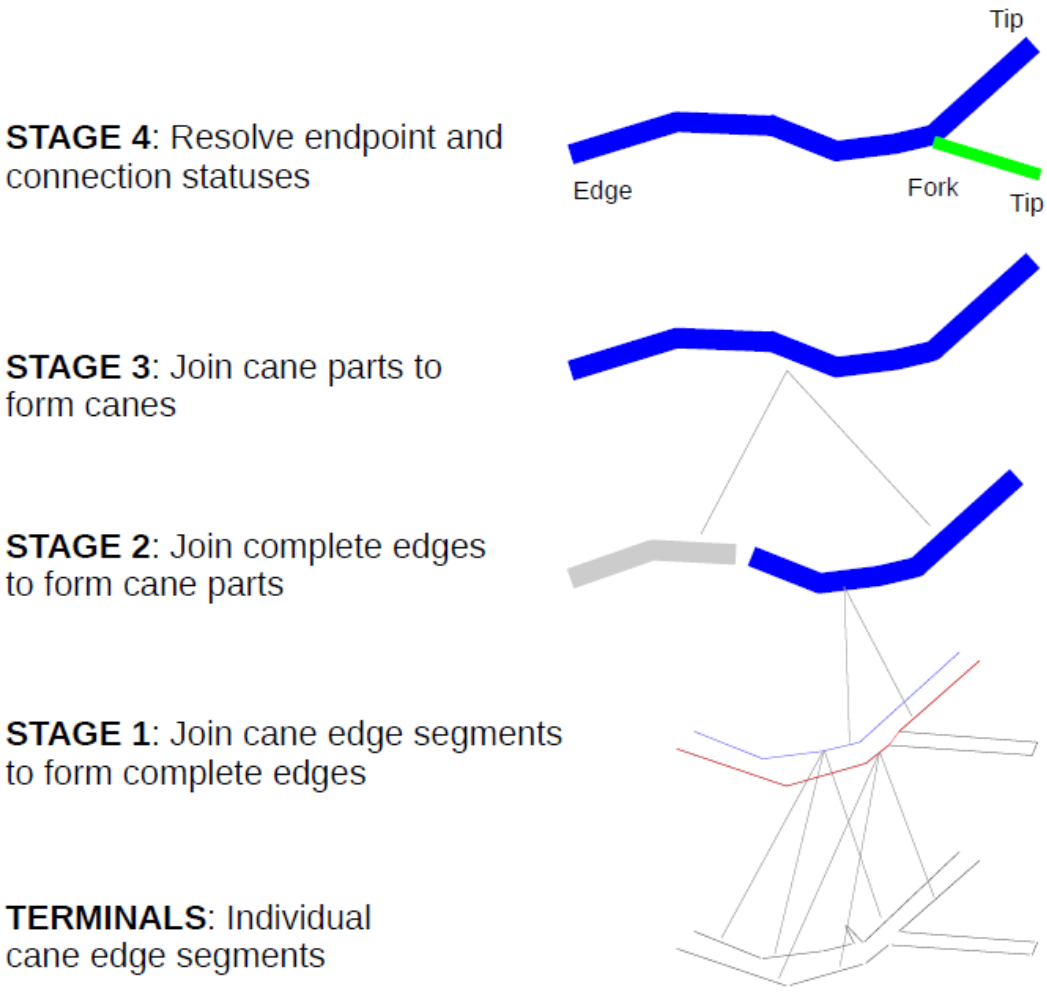


Figure 2.15: Bottom-up parsing of vine canes. Adapted from Botterill et al. [8].

Random Fields – MRF with parsing and grammars. Kozinski et al. [44] mapped the grammar graph representation to a Markov network where energy minimization can be used for MAP estimation. The method also makes use of bottom-up object detection as cues for building energy potentials, and it is used for facades parsing. In contrast, Vo et al. [95] generate a Bayes network from a grammar of activities in parsing videos. The author used a custom message passing algorithm for performing exact inference of posterior probabilities of high level activities in context. Finally, Botterill et al. [8] showed results of using bottom-up parsing of cane segments for vine structure extraction. This is illustrated in Figure 2.15. Here, segments are extracted

by simplifying edge polylines using the Ramer-Douglas-Peucker algorithm [8]. The hierarchical bottom-up representation is carried from cane edges little segments, to full cane edges, then to cane parts, and finally to complete canes. The method also involves use of machine learning, in particular SVM, to help in deciding which two pairs of parts should be joined or not. This method however do not use any grammar rules for top-down proposals and is exclusively bottom-up. Furthermore, the method, though use well trained classifiers for deciding connecting cane segments at any level or the hierarchy, is heuristic in the sense that there is no guarantee that the method recovers the vine structure, and even, there is no definition of structure. Future research on this technique points into looking at alternative primitives, and making use of a top-down parsing system [8].

2.3 Summary

The literature presents several methods that have been used for modeling plant and tree structure. The goals for most of these methods are ones of reconstructing and/or synthesizing new instances of trees and plants with realistic appearance. Many methods similar to ours, use 2D images to gather different structural information about the imaged tree. However, as seen in this chapter, almost none of the existing methods solves the problem formulated in the introduction of this thesis, because either they use manual intervention which we want to avoid, they make assumptions about the input which differs drastically from our robot pruning system setup, or they simply are not applicable directly to extraction of vine structure from images. On the other hand, vine structure extraction methods such as Botterill et al. [8] can be considered as the basis of the research presented in this thesis. Furthermore, I show comparative results against this prior research later in the results chapter.

Finally, my proposed solution to the vine structure problem is very similar to bottom-up and top-down approaches. Indeed, I aim to estimate vine structure by segmenting the vine into small cane regions, and from them recover whole canes and their connection using probabilistic inference. As with most techniques presented in this chapter that analyze tree information from images, I use skeletonisation as a representation of the tree main axis with the purpose of solving branching connections

from 2D images. Similarly, to many methods in this chapter, Djikstra's shortest path algorithm can be adapted with skeletonisation to model cane segments as polylines. In the next chapter we describe in detail my proposed solution and all the systems components used for solving my problem

3

Proposed Vine Structure Model For Binary Vine Images

In this chapter I present the methods that I propose for modeling vine structure in 2D images. The pipeline of my approach is shown in Figure 3.1. It is subdivided into two main components. On the first component, I am focused on modeling and finding cane segments in the input binary image. On the second component I am concerned about modeling and inferring the hierarchical connectivity of these cane segments—the *structure* of the vine. This chapter presents only the proposed models for cane segments and vine structure. Inference methods are presented in Chapter 4.

In the first part of my system, cane segments are modeled by using skeletonisation of vine binary maps and grouping skeleton pixels based on connectivity and adjacency (see next subsection). Medial axis techniques are widely used in computer vision for many applications such as character recognition, articulated objects representation, and to recover the tree structure of plant roots and blood vessels[1] [53, 51, 30]. I found skeletonisation to be easy to implement and easy to use and adapt to the inference component of the system. Details and methods for modeling of cane segments using skeletonisation is described in Section 3.1.

[1] See also Section 2.1.1.

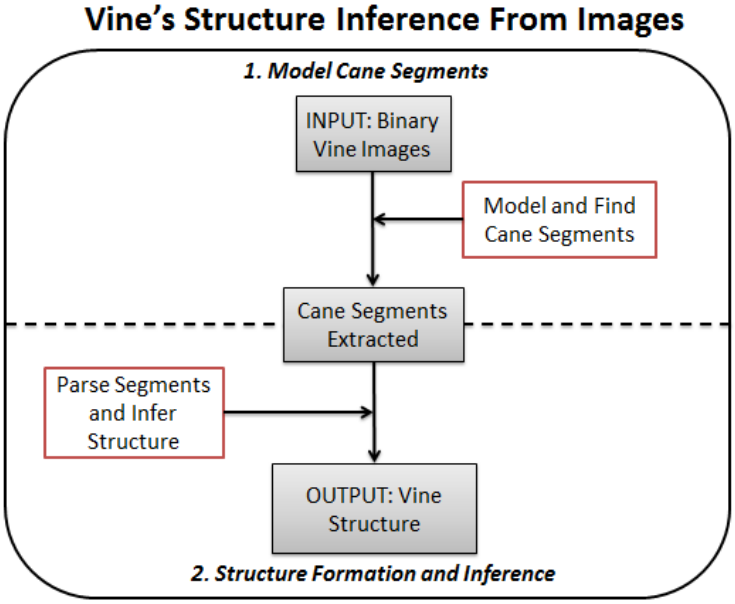


Figure 3.1: Pipeline of the proposed system for vine structure inference.

The second component of the system is about solving the occluded and branching vine regions from the extracted cane segments of the first component. It will recover the structure of the whole vine by specifying how the cane segments are connected to each other in a hierarchy. I propose to use a Hidden Markov Random Field for modeling vine structure, and to use Maximum A Posteriori (MAP) inference for solving the most likely structure for a vine image. MAP inference and in particular energy minimization techniques are highly used in computer vision for high level understanding of scenes and images, with methods like Part Based Models [27], Bottom Up/Top Down parsing [38] and Image Grammars/Parsing [93, 107]. A detailed description of my methods for modeling structure is presented in Section 3.2. MAP Inference and Energy minimization methods are presented in Chapter 4.

3.1 Modeling and Finding Cane Segments

I start by summarizing all names and terms I am going to use while talking about vine images. See Figure 3.2. A vine consists of many individual branches called *canes*. A cane grows from the root of the vine and may have many branching points

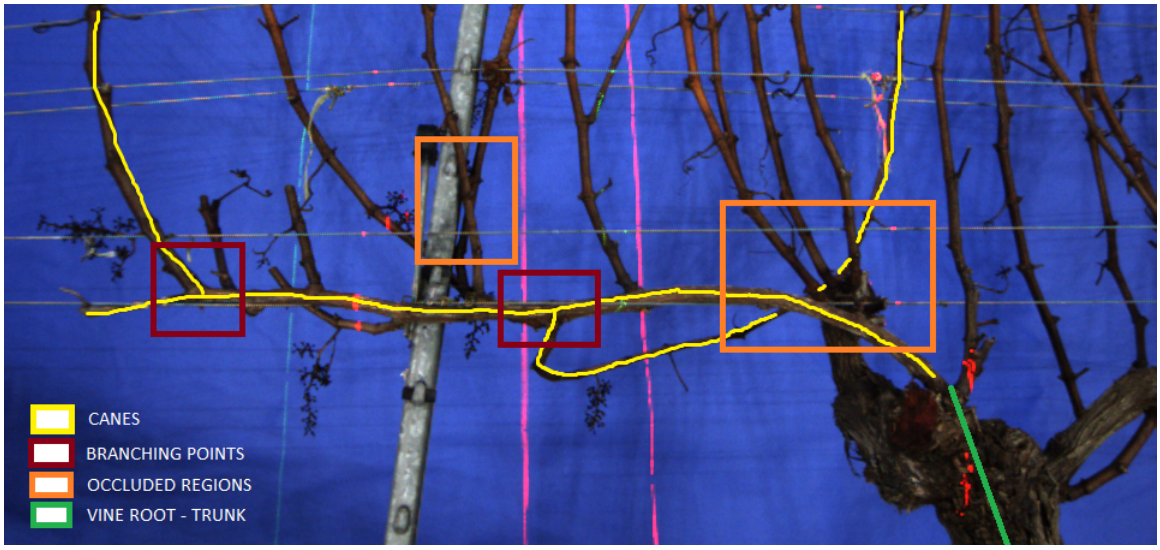


Figure 3.2: Vine nomenclature: Examples of canes are shown in yellow. A cane grows from the vine root, shown in green, or from other canes at branching points, shown in red. Some cane parts may be occluded by other canes and/or scene components. Examples of regions with occlusions and overlapping of canes are shown in orange.

from which other canes grow. In vine images, canes can be occluded by other canes or other scene components like the gray post in Figure 3.2.

The first part of my system is about modeling and finding vine canes in images. To model a cane of a vine, I used a polyline with thickness. A polyline is a natural representation for tree branches, and is also being used in the other parts of the robot pruning system. It also gives an order to the skeleton points of a single cane segment. The thickness measure will turn useful when solving connections between the modeled canes. On the other hand, finding full canes in vine images is hard to do directly, given the complexity of overlapping and occluded regions. See Section 2.1 for related research in reconstructing tree branches and similar subjects in images. Therefore, inspired by bottom-up/top-down methods, in my approach I further divide each cane into parts I call *cane segments*, so full canes can be reconstructed by solving connections between them. See Figure 3.3. A cane segment is still modeled as a polyline, but now the start and end points of the polyline may be referring to a cane starting point, a branching point, a cane end point, or a point where occlu-

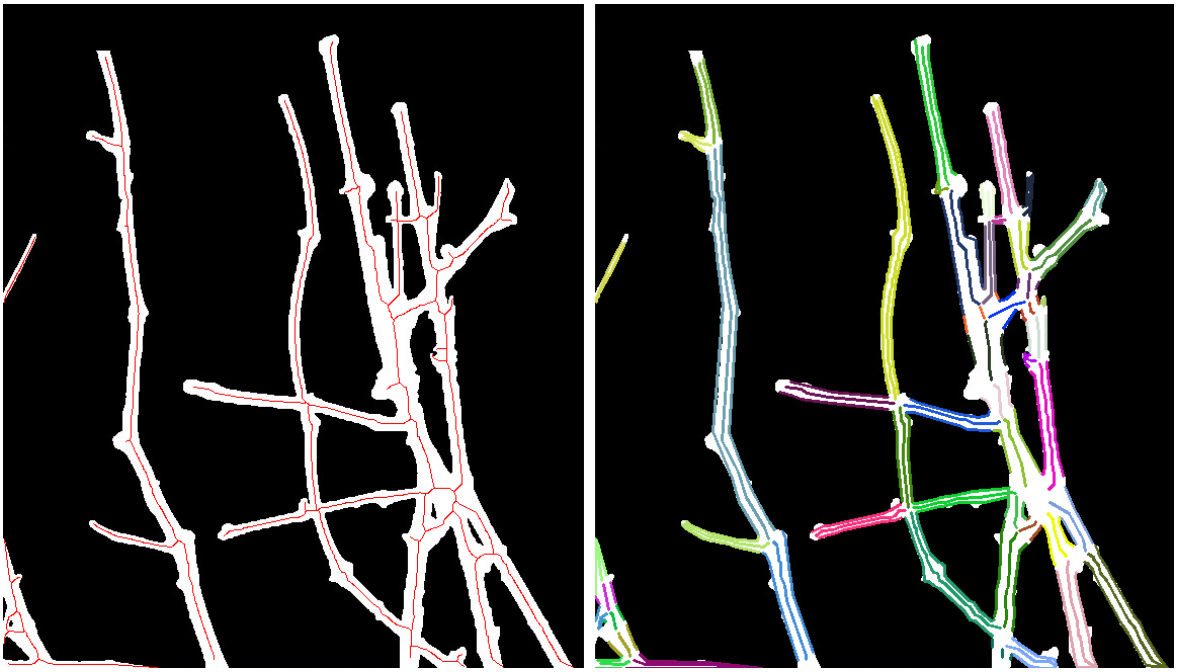


Figure 3.3: Skeleton Cane Segments: The skeleton of the vine image (left) is decomposed into polyline cane segments (right). Here different colors represent different cane segments.

sion/overlapping starts or ends. To find all polylines with thickness corresponding to a set of cane segments for an input vine image I used skeletonisation. In the next Section I explain this process in detail.

3.1.1 Extracting Cane Segments Using Skeletonisation

My method for finding cane segments is based on skeletonisation. See Figure 3.3. Here a skeleton image is decomposed into several polylines by splitting the skeleton points at junctions—points with more than 2 skeleton neighbors.

In my experiments I found that pruning of redundant small skeleton polylines can be done by just thresholding the length of the polylines. That is, if a polyline has length less than l pixels then the polyline is discarded. I used $l = 10$ pixels for my results. Observe also that each polyline is a list of connected skeleton pixels. Therefore, I simplified the polylines by using the Ramer-Douglas-Peucker algorithm with parameter $\epsilon = 1.0$. This let me get skeleton polylines with fewer control points,

while maintaining the shape of the original polyline unchanged [81].

One thing to note here is that I am using constant values of $l = 10$ and $\epsilon = 1.0$. Given that all input images we have in our system are all taken at a similar distance, I found these values heuristically. In the future, the choosing of these values can be studied in relation to the input binary maps at different scales and comparing the resulting set of segments to the ground truth number of segments. In this thesis the mentioned values were found to work well for the purpose of removing small segments in our particular type of input images.

Finally, with all polylines simplified, I assign to each polyline two edge contours. These edges will carry the thickness information of the cane model. A detailed description on how to compute thickness from these two edge contours is presented later in Section 3.3.3. The assignment of edge contours is done by iterating over all points of the polyline, and for each point selecting the two closest edge points that are in different sides of the polyline curve. Assigning a skeleton point to and edge point can be computed at the same time while computing the skeleton [72, 86, 90]. The problem with these approaches is that there is no way to control where the edge points are going to appear, and thus the property of the points in different sides of the polyline curve may not be easily satisfied. Figure 3.3 shows the final cane segments obtained for a vine binary image.

3.2 Introduction to Vine Structure Modeling and Inference

The second part of my system is about solving connections between the cane segments found in Section 3.1. The correct set of connections between cane segments will be referred as the *structure* of the vine. Here I used the *Maximum a Posteriori*–MAP estimation framework to solve the most likely structure of an input vine image. In this approach, I am interested in maximizing the posterior $P(\boldsymbol{x}|\boldsymbol{z})$ over a target variable $\boldsymbol{x} = [x_1, x_2, ..., x_m]$ given the observable data $\boldsymbol{z}$:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, P(\boldsymbol{x}|\boldsymbol{z}) \tag{3.1}$$

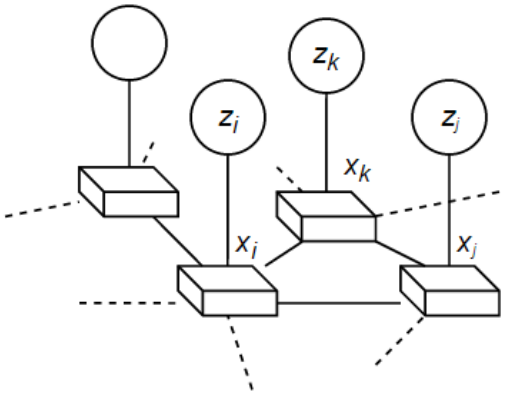


Figure 3.4: Graph in a Hidden Markov Random Field: Each node is associated with the hidden random variable x_i. The observable variables z_i provide evidence for particular values of x_i independently. This is shown as vertical edges. All other edges represent properties sharing, information passing or probabilistic bias between the two connected x's nodes [7].

The data $\boldsymbol{z}$ is also known as *evidence*. To relate this framework to our vine structure problem, I will be interested in solving the most likely state of connections between cane segments $\boldsymbol{x}^*$ given as evidence a set of measurable attributes $\boldsymbol{z}$ that I will define for each connection. Thus, $\boldsymbol{x}$ is the vector where each component x_i of $\boldsymbol{x}$ specifies if a connection between two cane segments is to be made or not. Also, each connection candidate x_i has an associated vector of attributes z_i that provide some evidence for whether the connection x_i should exist or not. This gives some intuition about the variables I will be using while talking about extracting the structure of vines using Equation 3.1. The formal definitions of $\boldsymbol{x}$ and $\boldsymbol{z}$ are presented in Section 3.3.

In my approach to structure inference, I researched the estimation of the posterior $P(\boldsymbol{x}|\boldsymbol{z})$ in the context of Hidden Markov Random Fields– HMRF. See Figure 3.4. A HMRF associated to $\boldsymbol{x}$ and $\boldsymbol{z}$ is build from an undirected graph, also known as a *Markov network* [7]. In this graph nodes are assigned the random variable x_i whose values are *hidden*– cannot be observed directly; and the evidence z_i are observations one can make at each node to explain the values of x_i independently. Also, edges are set between nodes that share information and/or properties. For example, in the HMRF created by taken each pixel of an image as a node, one may want neighbor

pixels to share the same property (e.g. color) and thus edges are constructed between neighboring pixels [48].

The motivation of using a HMRF for vine structure inference comes from considering cane segments connectivity decisions dependent to each other. Whether a cane segment is connected to another should be dependent on the state of other connections made for the cane. In this case, one benefit of a HMRF is that of catching implicitly the long-distance relations between x_i from the explicit local adjacency relations in the Markov network [7]. Furthermore, assuming all x_i satisfy the local Markov property of being conditionally independent of all others given its neighbors in the graph, the Hammersley-Clifford theorem [15], which applies to any positive probability distribution with the Markov property, allows to arrive to the following representation

$$P(\boldsymbol{x}|\boldsymbol{z}) = \frac{1}{Z(\boldsymbol{z})} e^{-E(\boldsymbol{x},\boldsymbol{z})} \tag{3.2}$$

where $Z(\boldsymbol{z}) = \sum_{\boldsymbol{x}} e^{-E(\boldsymbol{x},\boldsymbol{z})}$ is the normalization constant [2] that makes $P(\boldsymbol{x}|\boldsymbol{z})$ a valid distribution; and E is the energy function of the state $\boldsymbol{x}$ under observations $\boldsymbol{z}$. This representation of the posterior in Equation 3.2 is very convenient for the inference problem in Equation 3.1, because it translates the maximization of $P(\boldsymbol{x}|\boldsymbol{z})$ into the minimization of the energy function $E(\boldsymbol{x}, \boldsymbol{z})$:

$$\begin{aligned} \boldsymbol{x}^* &= \operatorname*{argmax}_{\boldsymbol{x}} P(\boldsymbol{x}|\boldsymbol{z}) \\ &= \operatorname*{argmax}_{\boldsymbol{x}} \ \log[P(\boldsymbol{x}|\boldsymbol{z})] \\ &= \operatorname*{argmin}_{\boldsymbol{x}} \ E(\boldsymbol{x}, \boldsymbol{z}) \end{aligned} \tag{3.3}$$

where I used the fact that $Z(\boldsymbol{z})$ does not depend on $\boldsymbol{x}$, and so does not need to be included during optimization. The energy function can be expressed as a sum of *likelihood* and *prior* terms [7, 48, 97]:

$$E(\boldsymbol{x}, \boldsymbol{z}) = \sum_i \Phi_i(x_i, z_i) + \sum_{c \in \mathcal{C}} \Psi_c(\boldsymbol{x}_c) \tag{3.4}$$

[2] $Z(\boldsymbol{z})$ is also known as the *partition function* [48].

where $\mathcal{C}$ is the set of *maximal cliques* in the underlying Markov network— defined as a set of subgraphs of the Markov network that are fully connected, such that adding any other node to a subgraph will spoil its fully connectedness[3] [97]. This expression is useful when modeling a problem using a HMRF, since it divides the energy into two terms that can be understood intuitively [7]. First a term that tell us how consistent/likely the current state variables x_i are according to the observations z_i— the unary potentials $\Phi_i(x_i, z_i)$; and second, a term that encapsulates the prior information one has about plausible states— the clique potentials $\Psi_c(\boldsymbol{x}_c)$.

The equivalence between MAP inference and energy minimization imply that, apart from methods that solve Equation 3.1 directly, many energy optimization frameworks and techniques that solve Equation 3.3 become available as well. In short, this thesis is about modeling and evaluating different approaches to recover structure of vines using Equations 3.1, 3.3 and 3.4. The following is a summary of the methods that I have considered for MAP estimation and the problem of extracting vine structures from images:

- **Iterated Conditional Modes**-ICM: this is a greedy iterative method for energy minimization. Given a current state solution, the method check all the possible modifications that can be done to this state, and chooses the one that achieves the lowest energy. The iteration finishes when no lower energies can be attained. Given the greedy nature of the algorithm, convergence is guaranteed only to a local minimum. This method is presented in Section 4.1.

- **Simulated Annealing**: this method is similar to ICM with two main differences. Firstly instead of checking all possible modifications to a state, the system propose a new state randomly. Secondly, the system can accept new states that increase the energy based on an acceptance probability controlled by a temperature parameter. Here, states that decrease the energy are always accepted. On the other hand, the lower the temperature, the lower the acceptance rate gets for states that increases the energy. This optimization framework is presented in Section 4.2.

[3] See Section3.3.5 and Figure 3.9 for more on cliques.

- **Gibbs Sampling based Random Search** : this is a method that make use of samples of the conditional distributions $P(x_i|x_1, ...x_{i-1}, x_{i+1}, ..., x_m, \boldsymbol{z}) := P(x_i|\boldsymbol{x}_{-i}, \boldsymbol{z})$, to reproduce samples from the posterior $P(\boldsymbol{x}|\boldsymbol{z})$. Sampling from the posterior is not the same as maximizing it, but still an estimation of the MAP can be taken as a function of the samples, e.g. the sample that reach maximum probability. This method is presented in detail in Section 4.3.3.

- **Min-Sum Loopy Belief Propagation**: this is a message passing procedure designed for energy minimization. Here, after initializing the *beliefs* at each node in the graph to 1, an arbitrary node is chosen to start propagating its information to its neighbors. The beliefs at each node are therefore updated iteratively when receiving and passing information using a min-sum formula. For graphs with loops, at every iteration, the inferred beliefs at each node are taken as approximations of min-marginal energies, however the message passing procedure may fail to converge. If convergence happens, an estimate of the MAP is build by using the estimations of the min-marginals of all nodes. This method is described in detail in Section 4.4.

Before going into each of these methods any further, in the next section I present the base definitions I will use for inference in the context of vine structures. That is, I will formally specify the HMRF model, including the state hidden variable $\boldsymbol{x}$, and the observations $\boldsymbol{z}$, a probabilistic model for $\boldsymbol{x}$ and $\boldsymbol{z}$, and an energy function according to Equation 3.4 and the chosen HMRF model. In the next chapter, I will describe in detail how to extract vine structures using all the methods above. The evaluation mechanisms and results for each method are presented in the Chapter 5.

3.3 HMRF Model for Vine Structure

In the present section I propose and define a HMRF model for vine structure inference. A HMRF model in the context of image processing and computer vision can be designed generally using the following ideas [7]:

1. Decompose a target image into a graph– *Markov network*; where nodes are

pixels or group of pixels, and edges are relations between them. Nodes and edges are defined depending on the problem one is solving.

2. Define a *hidden* random variable x_i at each node. This correspond to the information one is trying to infer, so it is not observable directly (e.g. a random variable to specify if a pixel is background or foreground).

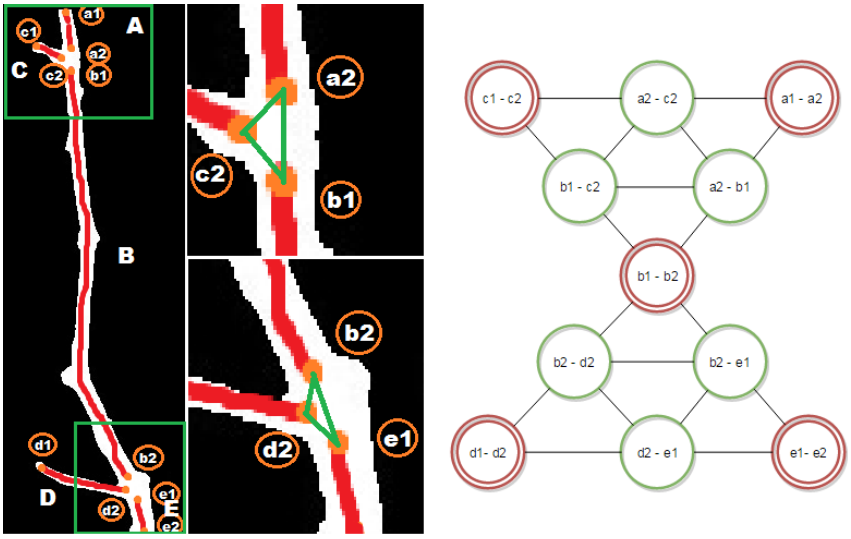
3. Define an *evidence* random variable z_i at each node. They give information about particular values of the hidden random variable associated to the same node. Therefore, evidence variables correspond to measurements one can make at each node (e.g. color information at a pixel).

4. Build a joint probabilistic model for both random variables x_i and z_i defined in the two previous steps.

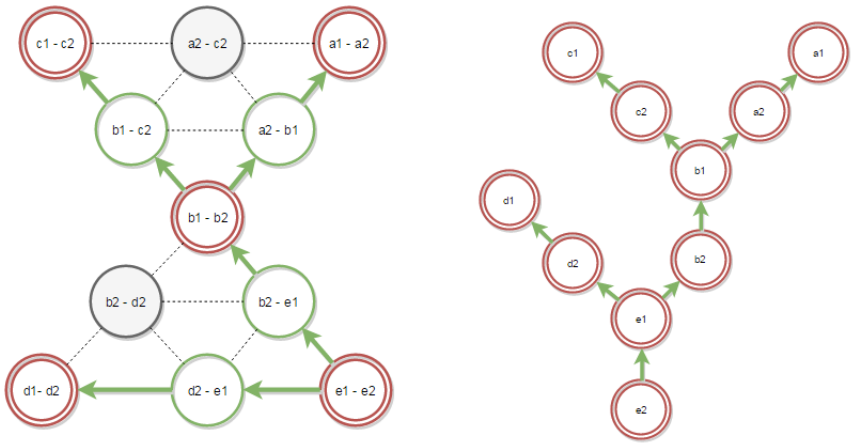
This section is structured in this same order. I start by defining a Markov network for my problem using the skeleton cane segments defined in Section 3.1. Then I define hidden variables x_i and evidence variables z_i related to vine structures. Finally, I present a probabilistic model and energy function that are my target for MAP inference as discussed in Section 3.2.

3.3.1 Markov Network of Cane Segments

My Markov model for vine structures is summarized in Figure 3.5. In the vine image shown there, cane segments polylines are shown in red and end points of the cane segments are shown in orange. The Markov graph in Figure 3.5 (a) is build by using as nodes *connections*. A connection is either a link between the two end points of the same cane segment (nodes in red), or a link between two end points of different cane segments (nodes in green). The notation for a connection between end points p and q is (p, q) and is symmetric in the sense that $(p, q) = (q, p)$. Edges are built from connections that share one end point in common. On the other hand, Figure 3.5 (b) illustrates how the vine structure graph looks like after inference. The vine structure graph is different from the Markov network. It is a tree, showing how



(a) Markov network of the vine structure model. On the left, cane segments are shown in red, with end points shown in orange. Connection candidates between the orange points are shown as green lines. On the right, a node in the Markov network correspond to a connection between either points in the same cane segment (double red circles), or points of different cane segments (single green circle). Edges in this graph are set on nodes that share a point in common (e.g. $e_1 - e_2$ is connected to $b_2 - e_1$ because they share the point e_1).



(b) Two different visualizations of how the Markov graph in (a) encodes the true hierarchical structure of the vine. The *flow* of connections of the vine is shown as green arrows. On the left, the flow of the vine growing is implicit, since one cannot see directly if the node $e_1 - e_2$ is oriented as $e_1 \to e_2$ or $e_2 \to e_1$. On the right the flow is explicit, by taking the visualization on the left and factoring out the green nodes and expanding each red node into single points.

Figure 3.5: MRF model for vine structure.

end points of the cane segments are connected and flowing from one cane segment to another.

Observe that the decision of using symmetry $(p, q) = (q, p)$ was taken because the way I build the network. The network has connections as nodes, and differentiating between (p, q) and (q, p) would add the double of nodes and thus several more edges to the graph. It is also easier to understand the network graphs when using symmetry. Also observe that symmetry is not a limitation, because we can still model directionality of a connection (that is, if a connection (p, q) is actually made as $p \to q$ or $q \to p$) by using a second binary variable as explained in the following section.

Given the set of skeleton cane segments, the procedure to build the Markov network is straight forward. The only thing that still needs to be specified is how to extract the set of connections between different cane segments, i.e, the green nodes in Figure 3.5 (a). A brute force approach is to consider as connections for a point, all points of other cane segments. However, I can simplify the number of connections (and then, the number of nodes in the Markov network), by considering for each end point, all end points of other cane segments that are at a fixed distance of D pixels as shown in Figure 3.6. This parameter D was chosen heuristically similar to the constants used in Section 3.1.1, i.e., they were found to be suitable for the particular type of vine binary maps that our system generates.

To finalize this section, note that given the skeleton cane segments, there are multiple ways in which I can model vine structure with a HMRF. For example, I could have chosen a model where nodes in the Markov network correspond to end points of the cane segments. However, I found this hard to handle in the following sense. Firstly, the hidden variables are harder to define when compared to my model. To see this, observe that in this alternative method, the hidden variables must be defined at each end point of all cane segments. A natural definition would be a random variable that specifies the label of another point to which it is connected. Given that the number of connections at a point is variable, this gets easily hard to describe. In contrast, the hidden variables in my method take boolean values that specify if a connection is to be made or not, and multiple connections are handled with no further modeling

(see Section 3.3.2). Secondly, defining the evidence variables z_i at each end point was not straightforward. The question here is, what kind of information can be measured at a point? Most HMRF models use color and/or similar measurements [7, 48]. However, this information either cannot be related directly to how cane segments in the presented model are connected or it is not enough for connectivity inference. For example, points with or without connections can have the same set of properties like color, or local angle, in which case the evidence would do a poor job in explaining when a connection should be made or not. On the other hand, the evidence variables in my model are specific to connections between cane segments. They can specify similarities or differences between two cane segments that are connected or unconnected, e.g. angle of connection or difference in cane thickness.

In summary, the presented model was chosen for its relative simplicity. Also, evidence variables z_i can easily be defined, and hidden variables x_i take boolean values that are straight forward to incorporate in terms of the energy function of Equation 3.4. In the following sections I present the formal definition of x_i and z_i. Details of the energy function are presented on Section 3.3.6.

3.3.2 Configuration of Connections Between Cane Segments – $\boldsymbol{x}$

In the previous section I constructed a Markov network where nodes are connections between end points of cane segments. To continue with the specification of my HMRF model for vine structure (see Section 3.3), in the present section I define the hidden variables x_i associated to each connection. The main idea is that instances of $\boldsymbol{x} = [x_1, x_2, ..., x_m]$ describe exactly which cane segments should be connected and which not. Also, I want to be able to describe directionality of connections, so I can infer the flow of growing of canes (see Figure 3.5 (b)). These characteristics can be achieved by using boolean states. The formal definition follows.

Refer to Figure 3.6. Denoting by $L(p)$ the set of end points that are candidates to be connected to end point p, I define the set of all pair of candidate connections

$$\mathcal{L} = \bigcup_{\forall p} L(p) = \bigcup_{\forall p} \{(p, q) |\ q \text{ is candidate to be connected to } p\} . \tag{3.5}$$

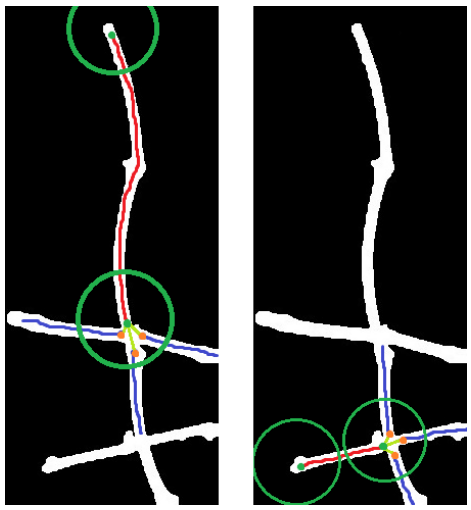


Figure 3.6: Examples of connection candidates. Here the candidates for the green points are the points shown in orange. These are found by casting a circle of fixed radius centered at the point. Note that some green points do not have associated any candidates at all.

Since (p, q) and (q, p) represent the same candidate connection, I remove these repetitions by keeping only one of these candidates. Therefore, in my model I have that $\mathcal{L}$ has no repetitions. As mentioned in Section 3.3.1, in my model the condition of q being candidate of point p is that of $\|p - q\| < D$, where D is measured in pixels (see Figure 3.6). Note also that $\mathcal{L}$ has a finite number of candidates. Let's write $\mathcal{L} = \{l_i\}$ with $i = 1, ..., m$. With this I can define for each element $l_i \in \mathcal{L}$ the random binary variable

$$j_i = j(l_i) = \begin{cases} 1 & \text{if connection } l_i \text{ exists} \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

In Figure 3.6, the light-green line segments from the green points to the orange points are examples of connection candidates l_i. Each l_i has the associated binary variable j_i that specifies if the corresponding cane segments should be connected or not. Thus, all j_i together can be used for specifying particular states of connections between all cane segments. This means different vine structure graphs can be achieved by toggling the values of j_i from connected (one) to unconnected (zero). This graph generated by all j_i however is incomplete in relation to the vine structure graph in Figure 3.5 (b), since the flow (direction) of the canes growing is missing,

that is, the information of whether a connection $l_i = (p_i, q_i) \in \mathcal{L}$ with $j_i = 1$, is a connection from p_i to q_i or from q_i to p_i. To account for this, for each l_i, I define a second random binary variable

$$f_i = f_i(l_i) = \begin{cases} 1 & \text{if } l_i : p_i \rightarrow q_i \\ 0 & \text{if } l_i : q_i \rightarrow p_i \end{cases} \tag{3.7}$$

where the arrow symbol $a \rightarrow b$ means the connection is flowing from a to b. Finally, after defining this directionality of connections, the hidden information for my HMRF is defined as a joint variable of connection states and directionality of all candidate connections l_i:

$$\begin{aligned} x_i &= (j_i, f_i) \\ \boldsymbol{x} &= [x_1, x_2, ..., x_m] \end{aligned} \tag{3.8}$$

I will call the hidden variable $\boldsymbol{x}$ a *configuration* of connections of all cane segments. The space of all possible configurations is denoted by:

$$X = \{[x_1, x_2, ..., x_m]\} = \{0,1\}^{2m} = \underbrace{\{0,1\}^2 \times \{0,1\}^2 \times ... \times \{0,1\}^2}_{\text{m times}} \tag{3.9}$$

Note that as I wanted at the beginning of this section, elements in the configuration space X, specify which cane segments are connected and which are not, including the direction of the corresponding connections. These configurations can be used for finding the most likely vine structure with MAP inference. Note also that for J join candidates the number of possible configurations is 4^J. Given that for a connection we only have three relevant states (one state of unconnected and two states of connected with different flows), the number of possible configurations for J join candidates can be thought as 3^J. In any case, since in average I have that $J > 100$, you can see why brute force search for the best configuration is not feasible.

3.3.3 Attributes for Connection Candidates – $\boldsymbol{z}$

In this section I define the evidence variables $\boldsymbol{z}$ for my vine structure HMRF model. This is done by defining a set of attributes $z_i = [\theta_i, w_i, d_i]$ for each candidate connection $l_i \in \mathcal{L}$, and then $\boldsymbol{z} = [z_1, ..., z_m]$. The attributes are angle θ_i, thickness difference w_i, and separation d_i between the points of connection l_i. These attributes in conjunction were found to be relevant to the decision of whether the connection l_i should exists or not, e.g. cane segments that have small thickness difference are more likely to be connected, but this condition by itself is not enough, so all three attributes are needed.

Despite all the methods presented in this thesis are related to binary images, I believe that color information could be an useful additional evidence for a connection. For this we need to assign to a color to a single connection (p, q) (node in our graph). This color can be chosen for example as the mean color of all line pixels between p and q, or as the difference between mean colors of the set of pixels of segment p and the set of pixels of segment q. In my methods however I left color information to be used in my future research, and here I concentrate on the three attributes of angle, thickness difference and separation. In the following subsections I describe how to compute each of these attributes for any given candidate connection l_i. A probabilistic model for z_i in relation to connection states $j_i(l_i)$ is described in Section 3.3.4.

Angle Attribute

To define the angle attribute θ_i, I first assign to each cane segment polyline $P = \{p_1, p_2, .., p_K\}$ a pair of directions v_{p_1}, v_{p_K} at its first and end points. Once this is done, I can define the angle attribute θ_i using the angle between the corresponding end points of the candidate l_i. Specifically, the angle between two end points p and q of the different polylines P and Q respectively, is defined as:

$$\theta_i(p, q) = cos^{-1}(\frac{v_p}{|v_p|} \cdot \frac{v_q}{|v_q|}) \tag{3.10}$$

To define the pair of directions v_{p_1}, v_{p_K}, look as a reference Figure 3.7, where

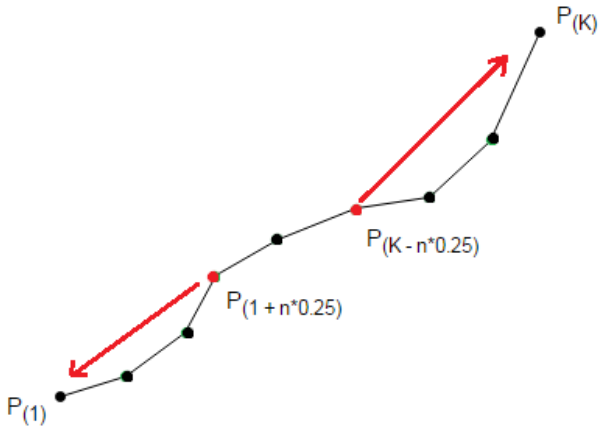


Figure 3.7: Computing polyline end point directions. Here n denotes the total number of points in the polyline. The directions at the endpoints of the polyline are shown in red. These directions are computed starting from the red points that are located at 25% before each of the end points, to the correspondent end point.

these directions are shown in red. The idea is to compute the direction between the start and end points the points (shown in red in the figure) that are located 25% ahead/behind in the polyline respectively. This percent threshold is constant and it was found heuristically to work well for our kind of images similarly as explained in Section 3.1.1. Mathematically, denoting by n the total number of points in the polyline, I compute these vectors as:

$$\begin{aligned} dir(p_{k_i}, p_{k_j}) &= \frac{p_{k_i} - p_{k_j}}{\left\| p_{k_i} - p_{k_j} \right\|} \\ v_{p_1} &= dir(p_1, p_{1+n*0.25}) \\ v_{p_K} &= dir(p_K, p_{K-n*0.25}) \end{aligned} \tag{3.11}$$

Alternative methods to define the directions v_{p_1}, v_{p_K} are, for example, the direction between the start and end points $v_{p_1} = dir(p_K, p_1) = -v_{p_K}$; and the extreme points directions $v_{p_1} = dir(p_2, p_1)$ and $v_{p_K} = dir(p_{K-1}, p_K)$. In my experiments I used the directions as computed in Equation 3.11.

Thickness Difference Attribute

I need to associate a thickness attribute to a connection $l_i = (p, q)$. I do this by first assigning a thickness value to a single cane segment. After this, I can assign the thickness w_i as the difference of the thickness of the cane segment of p and the thickness of the cane segment of q.

To find the thickness of a cane segment, recall from Section 3.1.1 that a cane segment is described by its center polyline and two edge polylines. Therefore I can use the two edges contours to approximate the mean width of a cane segment in pixels. I did this approximation in two main parts:

- **Build two edge contours for the cane segment so both of them have the same number of points :** This is done by construction at the stage of building the cane segments array. Refer to Section 3.1.1. The edge contours are built by iterating over the center polyline. For each center point p, I find two edges points on different sides relative to the center polyline. The easiest way of finding this two points is to have a precomputed edge image of the binary map, and find the two closest points of p to the edge map (distance measured in pixels). Here I use the direction of the polyline as the reference to know where the point is positioned relative to the center. Different from the direction computed in the previous section, in this construction we use a local direction between the previous point to p and p (when p is the last point) or p and the next point to p (when p is not the last point) in the cane segment polyline array. I used this local direction because the cane segment can have portions of both edge contours lying on the same side when using the global direction as computed in the previous section, and thus or edge construction rule would fail.

- **Compute the mean difference of distance between points that have the same index in the contours array :** Given the two edges arrays $\{e_{11}, e_{12}, ..., e_{1n}\}$ and $\{e_{21}, e_{22}, ..., e_{2n}\}$, the thickness of the corresponding cane segment s is approximated as :

$$thickness(s) = \frac{1}{n} \cdot \sum_{i=1}^{n} |e_{1i} - e_{2i}|$$

where we used the characteristic of our construction that both edges have the same number of points.

Separation Attribute

In this section we are interested in assigning a distance attribute to a connection (p, q). The simplest thing and natural way of defining this would be to assign the euclidean pixel distance between points p and q. However I found heuristically that a simple euclidean distance like this is not enough evidence for a connection (p, q) to be in neither state of connected or unconnected. This is because at branch overlapping regions we usually have several points close (by only few pixels) to each other, and thus this would bias our connections to connect all of the candidates at this branch overlapping. Also we do not want to penalize branches that are too far, because this would result in thresholding to define what is close or far, and thus would not generalize well. Furthermore, we want to consider to connect segments that may not be in the same connected component in the binary map, as the remotion of posts and wires can lead to input images with gaps and holes on single canes. Therefore, I my methods I defined the distance attribute in terms of the separation of a point to the line of the other point cane segment. This intuitively penalizes a lateral dislocation or "separation" of one cane segment to the other, while not penalizing if the two cane segments are close or not. Two important observations to make here are firstly, that our construction of candidate connections (p, q) already uses a localization and thresholding of cane segments (refer to Section 3.3.2) and therefore the separation attribute is never computed between two points of cane segments that are further apart than that threshold; and secondly, that this attribute is not implicitly covered by the angle attribute since for example two parallel cane segments are in agreement to the angle attribute, but can be dislocated at their candidate points from each other. In the following I explain in detail the computation of this separation attribute.

Given two end points of different polylines, I can compute the distance of one of the points to the line in the direction of the other point polyline. In Figure 3.8 this would be the distance of point p to the green line, and the distance of point q to the blue line, both distances shown in brown.

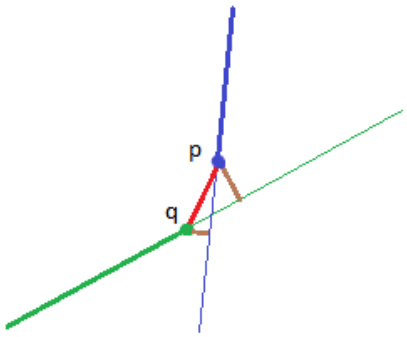


Figure 3.8: Computing separation between end points p and q of two polylines. The separation attribute for both points is defined as the averaged magnitude of the two brown segments. Each brown segment represent the distance of one point to the line in the direction of the other point. Directions of end points are computed as in Section 3.3.3.

Mathematically, I denote the line with direction v_q of q by $\lambda_q = q + t \cdot v_q$ for real values t. Then I denote by $d(p, \lambda_q)$ the distance of p to the line λ_q. With this notations I can compute:

$$d^2(p, \lambda_q) = |p - q|^2 - \langle (p - q), v_q \rangle^2 \tag{3.12}$$

The direction v_q is computed as in Section 3.3.3. The separation attribute d_i for end points p and q is just the averaged sum:

$$d_i(p, q) = \frac{1}{2}(d(p, \lambda_q) + d(q, \lambda_p)). \tag{3.13}$$

3.3.4 Probabilistic Model For $\boldsymbol{x}$ and $\boldsymbol{z}$

In the previous sections I have defined a Markov network for connections between cane segments (Section 3.3.1), hidden variables $\boldsymbol{x}$ that specify a configuration of connectivity (Section 3.3.2) and evidence variables $\boldsymbol{z}$ that characterize pairs of connected and unconnected cane segments (Section 3.3.3). To finalize the definition of my vine structure HMRF model, in this section I present a probabilistic model for $\boldsymbol{x}$ and $\boldsymbol{z}$.

Remember that my ultimate goal with the HMRF model presented so far in the previous subsections, is to perform MAP inference using Equation 3.1. Alternatively, as I showed in Section 3.2, this is equivalently to perform energy minimization with Equations 3.3 and 3.4. This means that the probabilistic model for $\boldsymbol{x}$ and $\boldsymbol{z}$ mentioned in this section, refers to either modeling the posterior distribution $P(\boldsymbol{x}|\boldsymbol{z})$ or the energy objective function $E(\boldsymbol{x}, \boldsymbol{z})$. For modeling the posterior I chose a generative model where prior information on $\boldsymbol{x}$ can be used. On the other hand, for energy minimization, I use the probability models specified for $P(\boldsymbol{x}|\boldsymbol{z})$ to define the energy potentials $\Phi_i(x_i, z_i)$ and $\Psi_c(\boldsymbol{x}_c)$. In the following I present in detail models for both posterior and energy functions that I will use for vine structure extraction. Evaluation of these models together with the different MAP optimization algorithms listed at the end of Section 3.2 are presented in Chapter 4.

3.3.5 Model for the Posterior $P(\boldsymbol{x}|\boldsymbol{z})$

Modeling the posterior $P(\boldsymbol{x}|\boldsymbol{z})$ can be done with either discriminative or generative models [68]. In a discriminative model, the posterior is defined directly and it usually requires supervised learning. Here I use a generative approach where, using prior information $P(\boldsymbol{x})$, one can use Bayes formula to model the posterior indirectly [7]:

$$P(\boldsymbol{x}|\boldsymbol{z}) \propto P(\boldsymbol{z}|\boldsymbol{x})P(\boldsymbol{x}) \tag{3.14}$$

where the constant of proportionality satisfies $\sum_{\boldsymbol{x}} P(\boldsymbol{x}|\boldsymbol{z}) = 1$ and does not depend on $\boldsymbol{x}$. Thus, instead of modeling the posterior directly, one can model separately the likelihood $P(\boldsymbol{z}|\boldsymbol{x})$ and the prior $P(\boldsymbol{x})$. Furthermore, as many Computer Vision methods that use MAP inference do [7, 48], I assume observations z_i are only dependent on its state x_i. This is intuitively justified for my data z_i since for example, the angle measured at a connection candidate l_i should not be dependent on any other connection candidate l_k, $k \neq i$, and so on with the rest of the attributes of thickness difference and separation. This assumption of independence of observations across connection candidates implies that the likelihood term $P(\boldsymbol{z}|\boldsymbol{x})$ can be easily

decomposed as the product:

$$P(\boldsymbol{z}|\boldsymbol{x}) = P(z_1|x_1)P(z_2|x_2)\cdots P(z_m|x_m) \tag{3.15}$$

which means that for computing the likelihood $P(\boldsymbol{z}|\boldsymbol{x})$, I just need to model generically a single $P(z_i|x_i)$ for any $i = 1, ..., m$, and evaluate the product in Equation 3.15. Similarly, in vision applications that make use HMRF models, the prior term $P(\boldsymbol{x})$ is usually decomposed as a product of factors [97, 7]

$$P(\boldsymbol{x}) \propto \prod_{c\in\mathcal{C}} \Psi_c(\boldsymbol{x}_c) \tag{3.16}$$

where $\mathcal{C}$ is the set of maximal cliques in our Markov network, and the proportional constant is the partition function (see Section 3.2). Therefore, our prior model can be factored into local prior information $\Psi_c(\boldsymbol{x}_c)$ of connection candidates $\boldsymbol{x}_c$ that are part of the same clique. In the following I present models for both $P(z_i|x_i)$ and $\Psi_c(\boldsymbol{x}_c)$ in the context of vines.

A Model for $P(z_i|x_i)$

In this subsection I am interested in defining the likelihood term $P(z_i|x_i)$ in order to be used in Equation 3.15. For this, first remember that in my HMRF model, the state variables $x_i = (j_i, f_i)$ are made from binary random variables of connectivity state j_i and flow directionality f_i (see Equation 3.8). Now, observe that in my model $P(z_i|x_i) = P(z_i|j_i)$. This is because the measurements of angle, thickness difference and separation are independent of the directionality of a connection, and so z_i is independent of flow f_i. Therefore, I only need to model $P(z_i|j_i)$. For this purpose, I specifically used two normal distributions for the binary values of j_i of connected ($j_i = 1$) and unconnected ($j_i = 0$) connection states:

$$\begin{aligned} P(z_i|j_i) &= [\mathcal{N}_0(z_i;\mu_0,\Sigma_0)]^{1-j_i}[\mathcal{N}_1(z_i;\mu_1,\Sigma_1)]^{j_i} \\ \mathcal{N}_v(z_i;\mu_v,\Sigma_v) &= \frac{1}{(2\pi)^{3/2}\sqrt{|\Sigma|}}\exp[-\frac{1}{2}(z_i-\mu_v)^T\Sigma^{-1}(z_i-\mu_v)]; \quad v = 0, 1. \end{aligned} \tag{3.17}$$

This is commonly known as *class conditional density functions* [68], since the density of each class v is modeled separately by a respective distribution, which is a normal in this case. The parameters of mean μ_v and covariance matrices Σ_v for each class can be learned in a supervised way. For this, I had manually annotated sets of connected ($v = 1$) and unconnected ($v = 0$) candidate connection samples: $S_v = \{z_1^v, z_2^v, .., z_{k_v}^v\}$. With these sample sets, using maximum likelihood estimation I have that

$$\begin{aligned} \mu_v &= \frac{1}{k_v} \sum_{k=1}^{k_v} z_k^v \\ \Sigma_v &= \frac{1}{k_v - 1} \sum_{k=1}^{k_v} (z_i^v - \mu_v)(z_i^v - \mu_v)^T \end{aligned} \tag{3.18}$$

Note that as usual in a generative model, I can use Bayes formula to compute the posterior of a single j_i:

$$P(j_i = 1|z_i) = \frac{P(j_i)P(z_i|j_i)}{P(j_i)P(z_i|j_i = 1) + (1 - P(j_i))P(z_i|j_i = 0)}. \tag{3.19}$$

This is exactly the model that I used in [57] together with Gibbs sampling for inference of vine structure. It models the probability of a single connection state given the set of attributes measured for that state. In my methods I set the prior $P(j_i) = 0.5$, so that with no other information, a connection j_i has the same probability of being set to connected or not.

A Model for $\Psi_c(\boldsymbol{x}_c)$

In this subsection I am interested in defining the clique factors $\Psi_c(\boldsymbol{x}_c)$ to be used in Equation 3.16. For this, first observe that these functions are independent of data attributes $\boldsymbol{z}$. Each factor carries only prior information of connection state j_i and flow directionality f_i in a maximal clique $c \in \mathcal{C}$ of the Markov network of the HMRF model (see Section 3.3.1 and Figure 3.5 (a)). Therefore, this subsection has two main objectives. First to specify the cliques of the Markov network of my model; and second to define explicitly Ψ_c in terms of j_i and f_i.

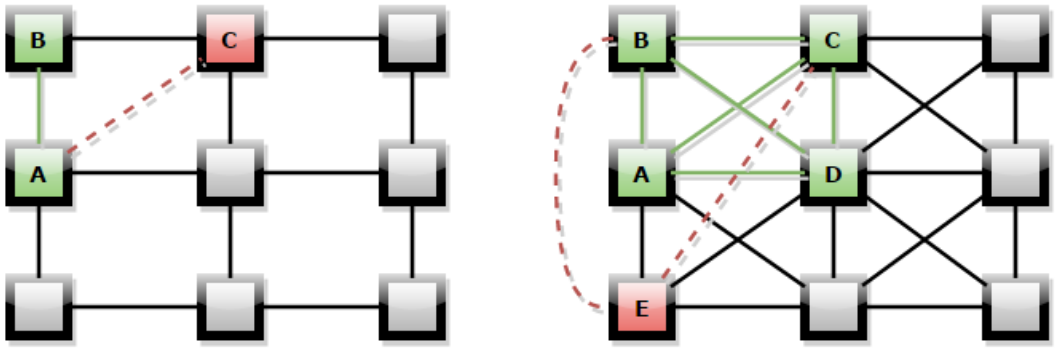


Figure 3.9: Two common graphs for Markov models in computer vision [7]. Examples of maximal cliques are shown in green. Adding one more (red) node to these maximal cliques would result in violating the clique connectivity property, since the red dashed connections are not part of the graph.

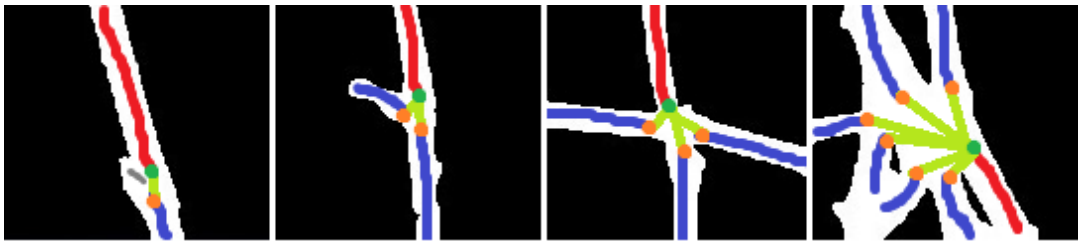


Figure 3.10: Maximal cliques in the cane segments Markov network presented in Section 3.3.1. In this image we see examples of maximal cliques the my vine model. As seen here the model can have maximal cliques of several orders. From left to right, we see examples of cliques of order two, three, four and seven. In any of these examples, the maximal clique for the connection in red is the collection of all connections (light green) that share a common point (dark green point) with the red connection. The gray line in the most left image correspond to a skeleton curve that was discarded during the pruning process (see Section 3.1).

I start defining cliques and maximal cliques. A *clique* of an undirected graph is a subset of nodes, such that every two distinct nodes in this subset are adjacent in the graph. See as a reference Figure 3.9. This figure shows two common Markov networks used for pixels in an image [7]. The graphs are generated from 4-connected pixels (on the left) and 8-connected pixels (on the right). Observe that by default every connected pair of nodes are cliques in both graphs. For example $\{A, B\}$ is a clique in both graphs since A and B are connected in both graphs. However,

Figure 3.11: Examples of flow contradiction. Flow is shown in gray arrows. Contradictions correspond to connections whose flows are oriented towards the green point they share.

$\{A, B, C\}$ is a clique only on the right graph, since there is no connection between A and C in the left graph. A clique is called a *maximal clique* if there is no other clique that contains it. In Figure 3.9 maximal cliques are shown in green. On the left graph, maximal cliques are pairs of connected nodes, since adding any third node will always violate the connectivity of the clique definition. On the right graph, maximal cliques are sets of four adjacent nodes. Here, pair of nodes are still cliques but not maximal, e.g. $\{A, B\} \subset \{A, B, C, D\}$ and thus $\{A, B\}$ is not a maximal clique. Finally, the *order* of a clique is defined as the number of elements the clique has. For my vine model I denote a clique by $c = \{c_1, c_2, \ldots, c_n\}$, which is a list of indices of hidden variables x_i. In this way, the vector $\boldsymbol{x}_c = (x_{c_1}, x_{c_2}, \ldots, x_{c_n})$ represents the subset of hidden variables corresponding to clique c, and n is the order of the clique.

Before giving the explicit definition of the clique potential functions Ψ_c for my vine model, first I want to define the information that I want to be carried or captured by any of these functions. This information can be divided into two main properties that I want the vine structure to have— a *maximum number of connections* at an end point, and no *flow contradictions*. Firstly, the maximum number of connections at a single end point is penalized to be at maximum three, with one of them being a cane segment itself. This is done to simplify our model of connectivity, so a cane segment can branch into maximum two other canes segments. Mathematically, for a maximal clique $\boldsymbol{x}_c = (x_{c_1}, x_{c_2}, \ldots, x_{c_n})$, this penalization can be written as a function g_M

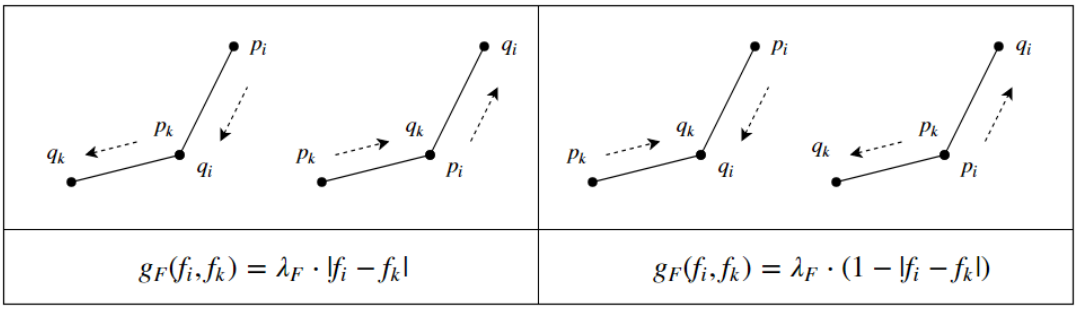


Figure 3.12: Flow penalty function g_F for two connection candidates $l_i = (p_i, q_i)$ and $l_k = (p_k, q_k)$. Dashed arrows represent the direction where $f_i = 1$ and $f_k = 1$. There are four possible configurations of l_i and l_j. The two cases on the left penalize differences in flow, while the two on the right penalizes equal flow values. For definition of flow see Section 3.3.2.

with penalty constant $\lambda_M > 0$:

$$g_M(\boldsymbol{x}_c) = \lambda_M \cdot [\max\{3, \sum_{i=1}^{n} j_{c_i}\} - 3]. \tag{3.20}$$

Observe that the sum term counts the number of connections candidates in the clique c that are set connected by setting $j_{c_i} = 1$, so if this number is less than our maximum number of connections allowed of three, then g_M is zero. On the other hand, when this number exceeds the limit, the penalty of λ_M is applied to the excess of number of connections.

Secondly, flow contradiction refers to connections that are flowing towards a point in common between the two connections. Examples of flow contradictions are shown in Figure 3.11. As before, I can penalize flow contradictions between two connections candidates $l_i = (p_i, q_i)$ and $l_k = (p_k, q_k)$ using the following function g_F with penalty constant $\lambda_F > 0$:

$$g_F(f_i, f_k) = \begin{cases} \lambda_F \cdot |f_i - f_k| & \text{if } q_i = p_k \text{ or } p_i = q_k \\ \lambda_F \cdot (1 - |f_i - f_k|) & \text{if } p_i = p_k \text{ or } q_i = q_k \end{cases}. \tag{3.21}$$

Note that there are four different cases of configuration between the two candidates l_i and l_k, leading to two different penalty terms. See as a reference Figure 3.12. In

the first two cases on the left of this figure, flow contradiction happens only when the flow values f_i and f_k are different. This is equivalent to $|f_i - f_k| = 1$. Similarly, in the last to cases on the right of the figure, flow contradiction happens when the flow values are equal. In this case $|f_i - f_k| = 0$, implying $1 - |f_i - f_k| = 1$. Note that regardless of the connections candidates l_i and l_k, the penalty function g_F throws a value of λ_F or zero. A compact formula for g_F can be achieved in terms of the *Kronecker delta* function δ:

$$\delta(a,b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases} . \tag{3.22}$$

With this I can write for g_F:

$$g_F(f_i,f_k) = \lambda_F\{(1-\delta(f_i,f_k))(\delta(q_i,p_k)+\delta(p_i,q_k))+\delta(f_i,f_k)(\delta(q_i,q_k)+\delta(p_i,p_k))\} \tag{3.23}$$

where both penalty terms have been summed and multiplied by equality functions between the end points of the connection candidates, and the terms involving $|f_i-f_k|$ have been rewritten in terms of δ as well.

After defining the penalty functions of g_M and g_F for maximum number of connections at a point and flow contradictions, I can now define the clique potentials Ψ_c for my vine model. First observe that in the examples shown in Figure 3.9 the maximal cliques have all the same order (pairs of nodes on the left, and 4-tuples of nodes on the right). In contrast, the Markov network of my vine structure model presented in Section 3.3.1, have maximal cliques with different orders. See Figure 3.10. As you can see there, maximal cliques in my model may have orders of two, three or more elements. This is because the number of connection candidates of cane segment end points vary along the vine image. Thus, in my model I considered defining clique potentials Ψ_c according to the order of the clique c on which the potential is defined. For this, first a pairwise potential is defined to penalize only flow contradiction:

$$\Psi_{ik}(x_i,x_k) = j_i j_k \cdot g_F(f_i,f_k) \tag{3.24}$$

Here there is no need to penalize the number of connections since in a clique of

order two there is only one connection that can be made. Also, the term $j_i j_k$ is added so flow contradiction penalties are only applied on connection candidates that are actually set to connected with $j_i = 1$ and $j_k = 1$. Now, for maximal cliques of order greater than two, $c = \{c_1, c_2, \ldots, c_n\}$, I define the clique potentials Ψ_c using the maximum number of connections penalty g_M and by gathering all possible sub cliques of c of order two $c_{ik} := \{c_i, c_k\} \subset c$ and using the definition of pairwise potentials of Equation 3.24:

$$\Psi_c(\boldsymbol{x}_c) = \sum_{c_{ik} \subset c} \Psi_{c_{ik}}(x_{c_i}, x_{c_k}) + g_M(\boldsymbol{x}_c) \tag{3.25}$$

Note that the pairwise potential in Equation 3.24 is symmetric with $\Psi_{ik}(x_i, x_k) = \Psi_{ki}(x_k, x_i)$. Therefore, in the sum of Equation 3.25, the combinations of sub cliques or order two of c are taken to be different with no repetitions, thus avoiding double penalization of the same flow contradiction.

Pairwise potentials are widely used in computer vision and they are possibly the most well researched models in HMRF methods given its relative simplicity and usefulness [7, 97, 68]. Also, inference algorithms like belief propagation turn out to be simpler when considering cliques of order two [7]. This justifies why I have chosen pairwise potentials for my vine model and structure inference.

3.3.6 Model for the Energy $E(\boldsymbol{x}, \boldsymbol{z})$

In this section I am interested in modeling an energy function $E(\boldsymbol{x}, \boldsymbol{z})$ in the form of Equation 3.4 with the purpose of performing energy minimization and vine structure inference. High energy values should be related to bad vine structures and connections, while the minimum energy value should be related to the most likely vine structure graph I can have for the set of cane segments. Remember that as shown in Section 3.2, the energy function can be modeled by specifying two terms of likelihood and prior information:

$$E(\boldsymbol{x}, \boldsymbol{z}) = \sum_i \Phi_i(x_i, z_i) + \sum_{c \in \mathcal{C}} \Psi_c(\boldsymbol{x}_c) := E_L(\boldsymbol{x}, \boldsymbol{z}) + E_P(\boldsymbol{x}) \tag{3.26}$$

This is in analogy to the model for the posterior $P(\boldsymbol{x}|\boldsymbol{z})$ presented in the previous section. The posterior was decoupled into likelihood $P(\boldsymbol{z}|\boldsymbol{x})$ and prior $P(\boldsymbol{x})$ terms factorized in Equations 3.15 and 3.16 respectively. Thus, for my vine model the energy potentials Φ_i and Ψ_c are directly defined using the likelihood and prior probabilistic models presented in the previous section. Specifically, for the prior term, the clique potentials Ψ_c that were defined in Equation 3.25 imply that:

$$\begin{aligned} E_P(\boldsymbol{x}) &= \sum_{c\in\mathcal{C}} \Psi_c(\boldsymbol{x}_c) \\ &= \sum_{c\in\mathcal{C}} [\sum_{c_{ik}\subset c} \Psi_{c_{ik}}(x_{c_i}, x_{c_k}) + g_M(\boldsymbol{x}_c)] \\ &= \sum_{c_{ik}\in\mathcal{C}} \Psi_{c_{ik}}(x_{c_i}, x_{c_k}) + \sum_{c\in\mathcal{C}} g_M(\boldsymbol{x}_c) \\ &= \sum_{c_{ik}\in\mathcal{C}} j_{c_i} j_{c_k} g_F(f_{c_i}, f_{c_k}) + \sum_{p\in\mathcal{Q}} g_M(\boldsymbol{x}_p) \\ &:= E_F(\boldsymbol{x}) + E_M(\boldsymbol{x}) \end{aligned} \tag{3.27}$$

Here I have used the fact that any clique of order two c_{ik} only appears one time in total. Also, I have denoted by $\mathcal{Q}$ the set of all end points p of all cane segments, and $\boldsymbol{x}_p$ the clique build from all candidate connections at point p. Therefore, the prior energy term E_P can be easily computed by summing the penalties of flow contradictions of every pair of cane segments connections E_F, and summing all the penalties for maximum number of connections made at end points E_M.

In turn, the likelihood potentials Φ_i are defined from the probability of connections given the observations $P(j_i|z_i)$:

$$\Phi_i(x_i, z_i) = j_i(1 - P(j_i|z_i)) + (1 - j_i)P(j_i|z_i) \tag{3.28}$$

The conditional probabilities $P(j_i|z_i)$ are computed with Bayes rule in Equation 3.19. Observe that each potential Φ_i can be understood as a penalty of a connection that was not made. The higher the probability of connection the less energy it will add when the connection $j_i = 1$ is set in the configuration. If the connection is not set, i.e $j_i = 0$, then it will add high energy values, unless the probability of connection

is small. This will guide energy optimization techniques to look into configurations with high probabilities of connections and keep unconnected those with low probabilities. Finally, the likelihood energy term can be written as:

$$\begin{aligned} E_L(\boldsymbol{x}, \boldsymbol{z}) &= \sum_i \Phi_i(x_i, z_i) \\ &= \sum_i j_i(1 - P(j_i|z_i)) + (1 - j_i)P(j_i|z_i) \end{aligned} \tag{3.29}$$

which means this energy term can be easily computed as the sum of probabilities of the connections that are made and missed in the configuration $\boldsymbol{x}$ with evidence $\boldsymbol{z}$.

This concludes the model for the vine structure. In the following sections I present the different algorithms for performing inference of the most likely vine structure. That is, methods for MAP inference and energy minimization.

4

Proposed Methods for Vine Structure Inference

The previous chapter presented a model for vine structure in binary images. This vine model consisted of two components. Firstly, a set of cane segment polylines which is extracted from images using skeletonization. Secondly, a HMRF that model connectivity and flow coherence between all cane segments. In this chapter I use different methods for performing inference of the most likely vine structure of a binary image, using the defined HMRF vine model. Evaluation and comparison of these inference methods are presented in the next chapter.

As seen in Section 3.2, MAP inference in HMRF models can be done analytically by maximizing directly the posterior distribution; by using adapting sampling techniques like Gibbs Sampling and Monte-Carlo methods; or by performing energy minimization. Analytical methods are suitable mostly for posterior distributions where a solution can be computed in closed form [7, 97]. Furthermore, MAP inference is known to be NP-hard in general [82], and except for specific types of graphs, the solutions found in non analytical methods would be just estimations of the true MAP values [41]. This is why, given the complexity of my HMRF vine model, I have chosen to evaluate and compare MAP estimations found using four different

non analytical methods. These four methods are described in this chapter with detail.

From the four methods that I have chosen for MAP estimation, Iterated Conditional Modes - ICM is the simplest approach. It is a greedy algorithm that is commonly used for evaluation of other MAP methods [87], and that is why I am including it in here. Similarly, simulated annealing is a non-greedy heuristic approach to energy minimization. This method is also considered mostly for comparative studies, where surprisingly often performs relatively better that other optimization techniques [78]. On the other hand, heuristic searches based on Gibbs sampling and Monte Carlo methods in general are a common choice for computer vision models that infer structure in images, like grammars and scene understanding [107, 93]. The method of Gibbs sampling is described together with the heuristic search algorithm applied to solve the vine structure inference problem. Finally, belief propagation and related message passing algorithms have been found recently to be highly accurate for MAP estimation [18, 87], and thus this method is a good candidate for solving my vine structure inference problem. In the following I present all four methods for MAP inference in the context of vine structures.

4.1 Iterated Conditional Modes-ICM

Iterated Conditional Modes-ICM is an iterative algorithm that can be used for MAP estimation and energy optimization. It was introduced by Besag in [4] in the context of image restoration using a MRF. The goal of this procedure is to obtain the MAP estimate $\boldsymbol{x}^*$ of Equation 3.1 by estimating each component as

$$x_i^* = \underset{x_i}{\operatorname{argmax}}\, P(x_i|z_i, \boldsymbol{x}_{c_i}) \tag{4.1}$$

where $\boldsymbol{x}_{c_i} = \{x_{c_1}, x_{c_2}, \ldots, x_{c_n}\}$ is the set of neighbors of x_i in the Markov network, i.e., the maximal clique on which x_i lies. The term $P(x_i|z_i, \boldsymbol{x}_{c_i})$ is called a *conditional mode*. The procedure of ICM refers to computing conditional modes for all x_i iteratively.

To relate this method to energy minimization, the conditional modes can be ap-

Algorithm 4.1 Iterated Conditional Modes - ICM

procedure ICM($\boldsymbol{x}_0, n_{max}$) ▷ Initial state and maximum number of iterations
  $\boldsymbol{x} \leftarrow \boldsymbol{x}_0$
  for $n \leftarrow 1, n_{max}$ **do**
    for $i \leftarrow 1, n_{comp}$ **do** ▷ iterate over the components of $\boldsymbol{x}$
      $E_{min} \leftarrow \infty$
      $x_{min} \leftarrow (f_0, j_0)$
      for $k \leftarrow 1, n_{state}$ **do** ▷ iterate over possible states of $x_i = (j_i, f_i)$
        $E_i \leftarrow$ EnergyCost($\boldsymbol{x}, i, k$) ▷ Equation 4.4
        if $E_i < E_{min}$ **then** ▷ minimize energy cost for x_i
          $E_{min} \leftarrow E_i$
          $x_{min} \leftarrow (j_k, f_k)$
        end if
      end for
      $x_i \leftarrow x_{min}$ ▷ update component i of $\boldsymbol{x}$
    end for
  end for
  return $\boldsymbol{x}$
end procedure

proximated in terms of likelihood and local conditional information [4]:

$$P(x_i|z_i, \boldsymbol{x}_{c_i}) = P(z_i|x_i)P(x_i|\boldsymbol{x}_{c_i}) \tag{4.2}$$

In this way, the analogous of ICM for energy minimization is to minimize iteratively a cost function of the value of a single component x_i, which is defined in terms of the likelihood and prior energy potentials Φ_i and Ψ_c respectively (refer to Section 3.3.6):

$$E(x_i) = \Phi_i(x_i, z_i) + \Psi_{c_i}(\boldsymbol{x}_{c_i}) \tag{4.3}$$

The ICM iterative procedure starts with an initial state $\boldsymbol{x}^{(0)}$. Then, in each iteration t, it updates each component $x_i^{(t)}$ while keeping all other components $x_k^{(t-1)}$, $k \neq i$ fixed. The update is done by optimizing the energy cost:

$$x_i^{(t)} = \underset{x_i}{\operatorname{argmin}}\, E(x_i, \boldsymbol{x}_{c_i}^{(t-1)}) = \underset{x_i}{\operatorname{argmin}}\, [\Phi_i(x_i, z_i) + \Psi_{c_i}(\boldsymbol{x}_{c_i}^{(t-1)})] \tag{4.4}$$

This process is repeated for all components until the energy cannot be decreased further, and thus all variables are assigned values that are locally optimal. Algorithm 4.1 presents an implementation of the ICM method. This procedure is guaranteed to find only a local optimum and is highly dependent on the initial state $\boldsymbol{x}^{(0)}$. A common approach is to run the algorithm several times using random initial states, and to choose the minimum energy state among all.

4.2 Simulated Annealing

Simulated annealing is a method that is suitable for combinatorial optimization. In this kind of problems, there exist an energy (objective) function to be minimized. This energy function is defined on a discrete and high dimensional *configuration* space [78]. The goal of simulated annealing is to find the configuration that minimizes the energy function. This minimization is done by exploring iteratively and randomly the configuration space. At a given iteration, the energy of the current configuration visited is evaluated and compared to the energy of the configuration in the previous iteration. Then, the system decides whether to accept or not the visited configuration. This decision is made by an acceptance probability in function of energy increases/decreases and a *temperature* parameter [78]. At high temperatures the system may accept increases on the energy, whereas at lower temperatures the system mostly accepts configurations that decrease the energy. Therefore, the exploration of the configuration space in simulated annealing is accompanied by a cooling mechanism from a starting high temperature. This enables the system to have a chance to escape local minimums, in contrast to greedy or gradient descent algorithms that only accept improvements on the configurations visited [78, 66]. Algorithm 4.2 presents an implementation of the Simulated Annealing process.

To make use of simulated annealing to a specific problem, there are 3 things that need to be specified [78]:

1. The configuration space X where the energy function is defined.

2. The energy function E to be minimized.

Algorithm 4.2 Simulated Annealing

```
1:  procedure SA(x_0, n_max, {t_1, ..., t_m})
2:      x ← x_0
3:      for T ← t_1, t_m do                          ▷ iterate over all temperatures
4:          for n ← 1, n_max do
5:              x' ← NEXTSTATE(x, T)                 ▷ jump to a new state randomly
6:              dE ← ENERGY(x') − ENERGY(x)
7:              if dE < 0 then
8:                  x ← x'                           ▷ accept always lower energy states
9:              else
10:                 p ← e^{−dE/T}
11:                 rnd ← UNIFORM(0, 1)
12:                 if rnd < p then
13:                     x ← x'                       ▷ accept with probability p
14:                 end if
15:             end if
16:         end for
17:     end for
        return x
18: end procedure
```

3. A mechanism to move from the state $\boldsymbol{x}^{(t)}$ to the next state $\boldsymbol{x}^{(t+1)} \in \eta(\boldsymbol{x}^{(t)})$, where $\eta(\boldsymbol{x}^{(t)})$ denotes the set of neighbors of the state $\boldsymbol{x}^{(t)} \in X$.

For my vine structure problem, the configuration space X and energy function E were defined in the HMRF vine model. Specifically, in Section 3.3.2 I defined configurations $\boldsymbol{x} \in X$ as an array of pairs of binary variables $x_i = (j_i, f_i)$ of connection states j_i and flow directionality f_i. In turn, the energy function E is explicitly defined on Equation 3.26 in Section 3.3.6. Therefore, the only missing component for using simulated annealing for inferring the most likely vine structure is that of the moving mechanism between states t and $t+1$, i.e, item number three of the list above. In the following I describe this moving mechanism in detail.

4.2.1 Moving Between Neighbor Configurations

To move from a state $\boldsymbol{x}^{(t)} \in S$ to a next state $\boldsymbol{x}^{(t+1)} \in \eta(\boldsymbol{x}^{(t)})$, there are two questions I need to address. Firstly, I need to define $\eta(\boldsymbol{x}^{(t)})$— the set of neighbors of

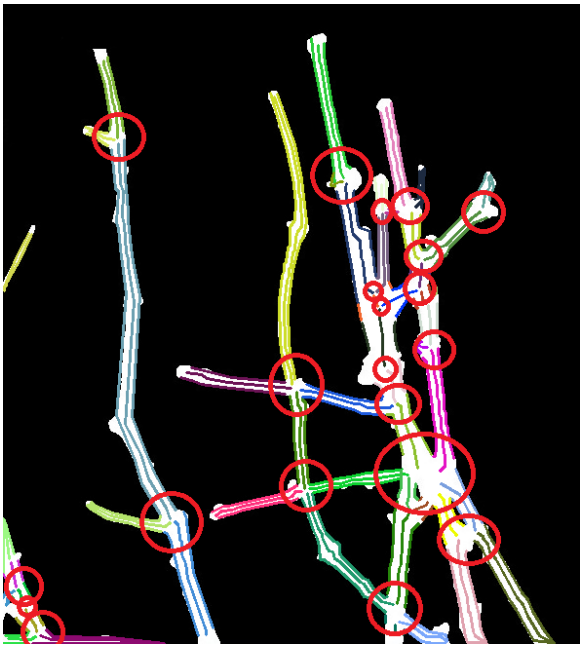


Figure 4.1: Grouping of candidates. Each circle represents a clique group of candidates. These are computed as candidates $l \in \mathcal{L}$ that share at least one end point.

$\boldsymbol{x}^{(t)}$ in X; and secondly I need to define how to actually choose one of these neighbors to be $\boldsymbol{x}^{(t+1)}$. In my approach, choosing $\boldsymbol{x}^{(t+1)}$ from $\eta(\boldsymbol{x}^{(t)})$ is simply a random choice among all elements of $\eta(\boldsymbol{x}^{(t)})$. Therefore, in this subsection I concentrate on defining $\eta(\boldsymbol{x})$ for any given state $\boldsymbol{x}$. This is done in the following two different ways.

Finding Neighbors by Shifting a Single Candidate

A simple way to define $\eta(\boldsymbol{x})$ is to simply pick up all states $\boldsymbol{x}' \in X$ such that $\boldsymbol{x} = [x_1, x_2, ..., x_m]$ and $\boldsymbol{x}' = [x'_1, x'_2, ..., x'_m]$ are the same except at a single component $i^* \in \{1, 2, \dots m\}$. I will refer to all $\boldsymbol{x}'$ that satisfy this condition, the *Single Shift* neighbors of $\boldsymbol{x}$.

Finding Neighbors by Shifting a Clique Candidates

A second approach to define $\eta(\boldsymbol{x})$ is to group together connection candidates $l \in \mathcal{L}$ that share at least one end point. That is, $\eta(\boldsymbol{x})$ is the maximal clique of $\boldsymbol{x}$ as defined in Section 3.3.1 and Section 3.3.4. See Figure 4.1. Observe that in this way,

I can have states $\boldsymbol{x}^{(t+1)}$ that are identical to the previous state $\boldsymbol{x}^{(t)}$ except in a small region of the vine. These small regions are shown in red circles in the Figure. I will refer to this kind of neighbors $\boldsymbol{x}'$ as the *Clique Shift* neighbors of $\boldsymbol{x}$.

4.3 Markov Chain Monte Carlo-MCMC

Another way to perform MAP estimation is by using Markov Chain Monte Carlo-MCMC sampling methods [24]. One of the main goals in this kind of sampling methods, is to generate a set of samples $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_r\}$ from a target probability distribution $P(\boldsymbol{x})$. Here $P(\boldsymbol{x})$ is complex enough that is hard to sample from it, and commonly it is only known up to a normalization constant Z [1]:

$$P(\boldsymbol{x}) = \frac{P^*(\boldsymbol{x})}{Z} \tag{4.5}$$

In my vine structure inference problem, the target distribution can be set to the posterior $P(\boldsymbol{x}|\boldsymbol{z})$ which was defined in Equation 3.2 and is equivalent to Equation 4.5. In Equation 3.2, I have $P^*(\boldsymbol{x}|\boldsymbol{z}) = e^{-E(\boldsymbol{x},\boldsymbol{z})}$ and the energy function is modeled as in Section 3.3.6. The motivation for using sampling methods for MAP inference is that by having samples of $P(\boldsymbol{x}|\boldsymbol{z})$ one can estimate $\boldsymbol{x}^*$ of Equation 3.1 using the sample that maximizes $P^*(\boldsymbol{x}|\boldsymbol{z})$, that is, the sample that minimizes the energy function $E(\boldsymbol{x}, \boldsymbol{z})$.

The need for different sampling mechanisms like MCMC , comes from the fact that sampling from a general probability distribution $P(\boldsymbol{x})$ is hard even if one can evaluate the unnormalized factor $P^*(\boldsymbol{x})$. Useful samples from $P(\boldsymbol{x})$ must correspond to states $\boldsymbol{x}$ where $P(\boldsymbol{x})$ is relatively high. However, for a vast state space X where $\boldsymbol{x} \in X$, there is no easy way to extract the regions of X where P is high, and brute force search would not be viable given the dimensionality of X. One can try using uniform sampling, but if P is densely concentrated in small regions of X, the amount of samples needed would again be not feasible. A source of examples that demonstrates this difficulty in sampling can be found in [56].

Equally to simulated annealing, MCMC methods explore the configuration space X iteratively. However, now the sequence of visited configurations $\boldsymbol{x}^{(t)}$ builds a

Markov chain. It has been shown that given the target distribution $P(\boldsymbol{x})$, one can construct a Markov chain satisfying certain conditions in order for the distribution of the samples $\boldsymbol{x}^{(t)}$ converge in the limit to $P(\boldsymbol{x})$ [24, 1]. This means that after sufficiently long number of iterations, the states $\boldsymbol{x}^{(t)}$ can be taken as samples of $P(\boldsymbol{x})$. In the following I present in detail the theory of MCMC sampling and in particular the Metropolis-Hastings and the Gibbs sampling methods for my vine structure problem.

4.3.1 Markov Chains and Detailed Balance

A Markov chain in a discrete and finite space X is defined using transition probabilities from state $\boldsymbol{x}$ to a new state $\boldsymbol{x}'$:

$$T(\boldsymbol{x}';\boldsymbol{x}) = P(\boldsymbol{x}'|\boldsymbol{x}) \tag{4.6}$$

with $\sum_{\boldsymbol{x}'} T(\boldsymbol{x}';\boldsymbol{x}) = 1$ for all $\boldsymbol{x}$, which means that for any given state $\boldsymbol{x}$, the transition probabilities of all states $\boldsymbol{x}'$ that can be reached from $\boldsymbol{x}$ sums to one. Now, denoting by $p^{(0)}(\boldsymbol{x})$ an arbitrary initial distribution on states $\boldsymbol{x} \in X$, the Markov chain specified by T induces a sequence of distributions $p^{(t)}(\boldsymbol{x})$ of states $\boldsymbol{x} \in X$ defined for $t > 0$ [1]:

$$p^{(t)}(\boldsymbol{x}') = \sum_{\boldsymbol{x}} T(\boldsymbol{x}';\boldsymbol{x}) \cdot p^{(t-1)}(\boldsymbol{x}) \tag{4.7}$$

The sequence $p^{(t)}$ in general depends on the starting distribution $p^{(0)}$. However, it can be shown that under certain conditions imposed on the Markov chain T (see below), for any initial $p^{(0)}$ the distributions $p^{(t)}$ converge to a *stationary* distribution of T [24, 14]. A probability distribution $\pi(\boldsymbol{x})$ is said to be stationary with respect to a Markov chain T, if it satisfies:

$$\pi(\boldsymbol{x}') = \sum_{\boldsymbol{x}} T(\boldsymbol{x}';\boldsymbol{x}) \cdot \pi(\boldsymbol{x}) \tag{4.8}$$

The conditions imposed on the Markov chain T in order for getting the convergence $\lim_{t\to\infty} p^{(t)} = \pi$ to a stationary distribution π of T, and for any arbitrary $p^{(0)}$ are [1]:

- *Irreducibility*: It is possible to get to any state $\boldsymbol{x}'$ from any other state $\boldsymbol{x}$.

- *Aperiodicity*: The chain must not get stuck in cycles.

To relate stationary distributions of T and convergence of $p^{(t)}$ to the problem of sampling from a target distribution $P(\boldsymbol{x})$, consider what happens when $P(\boldsymbol{x})$ is such that $\lim_{t\to\infty} p^{(t)} = P$ and P is stationary for T. While running the Markov chain T, at each iteration t, the state $\boldsymbol{x}^{(t)}$ is a sample from the distribution $p^{(t)}$. Thus, because of the convergence of $p^{(t)}$ to P, one will have that after enough iterations have been performed on the chain, lets say N iterations, the samples $\boldsymbol{x}^{(t)}$ will be approximately samples of P for all $t > N$.

In summary, the idea of MCMC methods is to design a Markov chain T satisfying the conditions of irreducibility and aperiodicity and such that the stationary distribution to which $p^{(t)}$ converges is the target distribution $P(\boldsymbol{x})$ from which one wants to sample. This designing of T may appear as hard as the original problem of sampling from $P(\boldsymbol{x})$. A simpler alternative, from the different methods that exist to design T (see [56]), is defining T such that the target distribution P satisfies the detailed balance equation [93, 7, 1, 56]:

$$P(\boldsymbol{x}')T(\boldsymbol{x};\boldsymbol{x}') = P(\boldsymbol{x})T(\boldsymbol{x}';\boldsymbol{x}) \tag{4.9}$$

It can easily be shown by summing both sides over $\boldsymbol{x}$, that if P satisfies Equation 4.9, then P satisfies Equation 4.8 and thus P is a stationary distribution of T [1]. A convenient property of Equation 4.9 is that because the normalization constants cancel out, one can design T by using the unnormalized factor $P^*(\boldsymbol{x})$ [14]. This is the case for my HMRF vine model that uses a Gibbs distribution with unknown normalization constant (see Equation 3.2). In the following sections, I describe two fundamental methods that specify T according to the detailed balance equation. The methods are Metropolis-Hastings and Gibbs sampling. These methods has been used before for image parsing and bottom-up/top-down scene understanding [93, 1, 7] and thus they are of importance for my evaluation of methods used for vine structure inference.

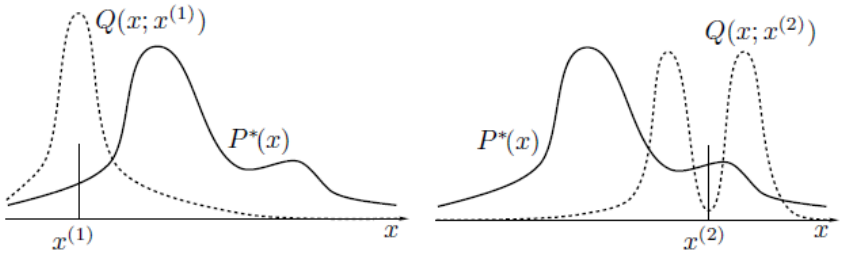


Figure 4.2: Metropolis-Hastings sampling from a proposal distribution $Q(\boldsymbol{x}'; \boldsymbol{x})$. The proposal distribution Q shown in dashed lines may be dependent on the current sample $\boldsymbol{x}$. Thus a sequence of points has associated a sequence of proposals. The target distribution $P^*(\boldsymbol{x})$ is shown in solid line. Adapted from [56].

4.3.2 The Metropolis-Hastings Method

The Metropolis-Hastings method makes use of a proposal distribution $Q(\boldsymbol{x}; \boldsymbol{x}^{(t)})$ that in general depends on the current state $\boldsymbol{x}^{(t)}$, and such that is easy to sample from it. See Figure 4.2. The method starts with an initial state $\boldsymbol{x}^{(0)}$ and iteratively, at each iteration t a new state $\boldsymbol{x}'$ is proposed as a sample of the current proposal $Q(\boldsymbol{x}, \boldsymbol{x}^{(t)})$. The new sample $\boldsymbol{x}'$ is accepted using the ratio [24]:

$$\alpha(\boldsymbol{x}'; \boldsymbol{x}^{(t)}) = \frac{P^*(\boldsymbol{x}')}{P^*(\boldsymbol{x}^{(t)})} \cdot \frac{Q(\boldsymbol{x}^{(t)}; \boldsymbol{x}')}{Q(\boldsymbol{x}'; \boldsymbol{x}^{(t)})} \tag{4.10}$$

If $\alpha(\boldsymbol{x}'; \boldsymbol{x}^{(t)}) \geq 1$ then $\boldsymbol{x}'$ is accepted, otherwise it is only accepted with probability $\alpha(\boldsymbol{x}'; \boldsymbol{x}^{(t)})$. Also if the new state $\boldsymbol{x}'$ is accepted then one sets $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}'$ otherwise one retains the previous state and set $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)}$. This iterative procedure can be seen as a MCMC method that defines the transition matrix T by:

$$T(\boldsymbol{x}'; \boldsymbol{x}) = \begin{cases} Q(\boldsymbol{x}'; \boldsymbol{x}) & \text{if } \boldsymbol{x}' \neq \boldsymbol{x};\ \alpha(\boldsymbol{x}'; \boldsymbol{x}) \geq 1 \\ Q(\boldsymbol{x}'; \boldsymbol{x}) \cdot \alpha(\boldsymbol{x}'; \boldsymbol{x}) & \text{if } \boldsymbol{x}' \neq \boldsymbol{x};\ \alpha(\boldsymbol{x}'; \boldsymbol{x}) < 1 \\ Q(\boldsymbol{x}'; \boldsymbol{x}) + \beta(\boldsymbol{x}) & \text{if } \boldsymbol{x}' = \boldsymbol{x};\ \beta(\boldsymbol{x}) = \sum\limits_{\alpha(\boldsymbol{y}; \boldsymbol{x}) < 1} Q(\boldsymbol{y}; \boldsymbol{x})(1 - \alpha(\boldsymbol{y}; \boldsymbol{x})) \end{cases} \tag{4.11}$$

It can be shown that this transition matrix T satisfies the detailed balance Equation 4.9 with respect to $P(\boldsymbol{x})$ [24, 9]. Therefore, as seen in the previous section,

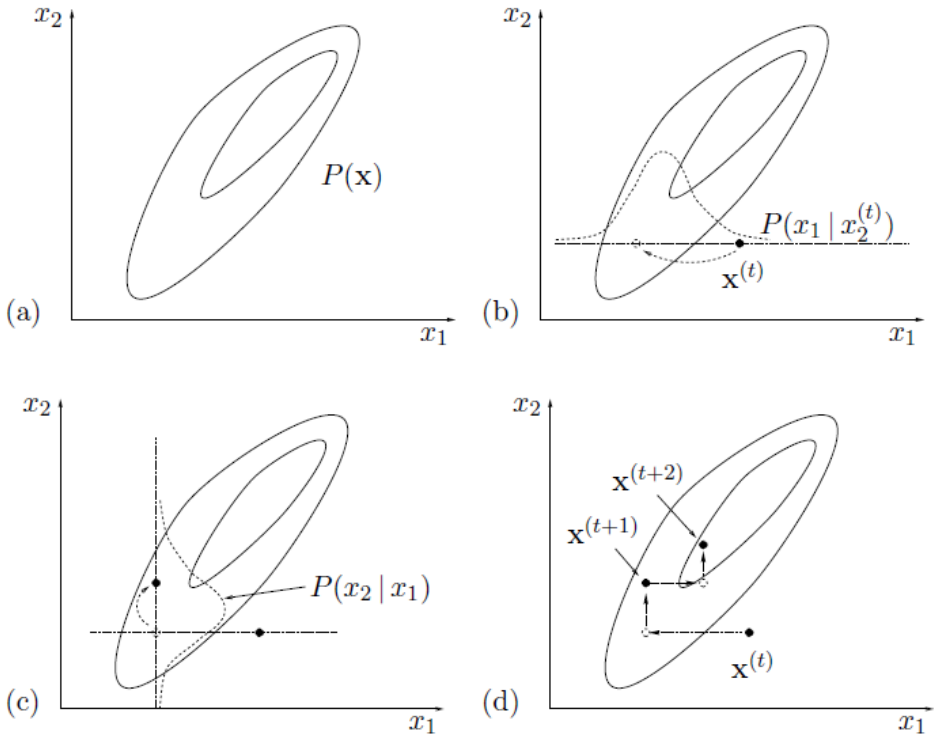


Figure 4.3: Gibbs sampling in a 2D state space (x_1, x_2). (a) the target distribution $P(\boldsymbol{x})$; (b) at iteration t a new $x_1^{(t+1)}$ is taken as a sample from $P(x_1|x_2^{(t)})$; (c) still at iteration t, a new $x_2^{(t+1)}$ is taken as a sample of $P(x_2|x_1^{(t+1)})$. (d) two iterations of Gibbs sampling. Adapted from [56].

the distribution of the samples $\boldsymbol{x}^{(t)}$ converges to $P(\boldsymbol{x})$. Furthermore, the convergence is independent on the chosen initial state $\boldsymbol{x}^{(0)}$. The proposal distribution Q can be defined as simple as a Gaussian, and when defined symmetric with $Q(\boldsymbol{x}'; \boldsymbol{x}) = Q(\boldsymbol{x}; \boldsymbol{x}')$ the acceptance probability reduces to the simple proportion $\alpha(\boldsymbol{x}'; \boldsymbol{x}^{(t)}) = \frac{P^*(\boldsymbol{x}')}{P^*(\boldsymbol{x}^{(t)})}$. This simplification is also known as the Metropolis method [14].

4.3.3 Gibbs Sampling

Gibbs sampling is a special case of the Metropolis-Hastings method seen in the previous section. In Gibbs sampling, we denote with $P(x_i|x_1, ...x_{i-1}, x_{i+1}, ..., x_m) := P(x_i|\boldsymbol{x}_{-i})$ the conditional distribution of a single component given all the others. Then, given a current sample $\boldsymbol{x}^{(t)}$, a new sample $\boldsymbol{x}'$ is constructed as identical to $\boldsymbol{x}^{(t)}$

Algorithm 4.3 Heuristic Search Based on Gibbs Sampling

procedure GS($\boldsymbol{x}_0, n_{max}$)
  $\boldsymbol{x} \leftarrow \boldsymbol{x}_0$
  $\boldsymbol{x}_{best} \leftarrow \boldsymbol{x}_0$
  $E_{min} \leftarrow$ ENERGY($\boldsymbol{x}_0$)
  for $n \leftarrow 1, n_{max}$ **do**
    for $i \leftarrow 1, n_{comp}$ **do** ▷ iterate over the components of $\boldsymbol{x}$
      $x_i \leftarrow$ CONDITIONALSAMPLE($\boldsymbol{x}, i$)
    end for
    $E \leftarrow$ ENERGY($\boldsymbol{x}$)
    if $E < E_{min}$ **then** ▷ keep track of the minimum energy state
      $E_{min} \leftarrow E$
      $\boldsymbol{x}_{best} \leftarrow \boldsymbol{x}$
    end if
  end for
  return $\boldsymbol{x}_{best}$
end procedure

except in the component i. Finally, the component i is sampled from the distribution

$$Q(\boldsymbol{x}'; \boldsymbol{x}^{(t)}) = P(x_i | \boldsymbol{x}_{-i}^{(t)}) \tag{4.12}$$

Note that by replacing Equation 4.12 in Equation 4.10 we have that in Gibbs sampling $\alpha(\boldsymbol{x}', \boldsymbol{x}) = 1$ and thus all new states $\boldsymbol{x}'$ are accepted. Figure 4.3 illustrates the iterative procedure of Gibbs sampling for a two dimensional target distribution $P(\boldsymbol{x}) = P(x_1, x_2)$. In practice and for more dimensions, at a single iteration t each component $x_i^{(t)}$ of $\boldsymbol{x}^{(t)}$ is updated sequentially by sampling from the conditionals on the already updated and non-updated components:

$$\begin{aligned}
&x_1^{(t+1)} \text{ sample from } P(x_1 | x_2^{(t)}, ..., x_m^{(t)}) \\
&x_2^{(t+1)} \text{ sample from } P(x_2 | x_1^{(t+1)}, x_3^{(t)}, ..., x_m^{(t)}) \\
&x_3^{(t+1)} \text{ sample from } P(x_3 | x_1^{(t+1)}, x_2^{(t+1)}, x_4^{(t)}, ..., x_m^{(t)}) \\
&\quad \vdots \\
&x_m^{(t+1)} \text{ sample from } P(x_m | x_1^{(t+1)}, x_2^{(t+1)}, ..., x_{m-1}^{(t+1)})
\end{aligned} \tag{4.13}$$

Gibbs sampling is useful in the case where $P(\boldsymbol{x})$ is complex, but the conditionals $P(x_i|\boldsymbol{x}_{-i})$ are easy to sample [56]. This turns out to be true for MRF models where the conditionals are reduced to conditionals on neighbor states, $P(x_i|\boldsymbol{x}_{-i}) = P(x_i|\boldsymbol{x}_{c_i})$ [7]. In the context of my inference problem, Algorithm 4.3 presents an adaptation of the Gibbs Sampling method to perform an heuristic search for the minimum energy state. The idea is basically to keep track of the minimum state at all iterations of normal Gibbs sampling, which we know that after long runs will resemble samples of our target distribution. Evaluation of this heuristic against all other methods is presented in the next Chapter.

4.4 Min-Sum Loopy Belief Propagation

The min-sum belief propagation algorithm is a message passing method that can be used for energy minimization [7]. Message passing algorithms operate on a target graph of a graphical model, a MRF graph or a Bayesian network [100]. Here I will use it for my HMRF vine model which has a corresponding graph described in Section 3.3.1. Now, there are two types of quantities involved in a message passing algorithm. The first quantity is that of *messages*, which are values that are passed from node to node in the graph. Intuitively, a node x_k tells to its neighbor node x_i what it believes about the plausible values that x_i should take. This message value $m_{k\to i}$ from x_k to x_i is computed as information that the source node x_k has gathered from itself and its other neighbors $x_{i'}$, but with $i' \neq i$ so there is no direct reinforcement of the information that x_i already has. Particularly to the min-sum algorithm for minimizing a target energy function with potentials Φ_k and Ψ_{ik}, messages between nodes are defined by the following formula borrowed from [7]:

$$m_{k\to i}(x_i) = \min_{x_k} \left(\Phi_k(x_k) + \Psi_{ik}(x_i, x_k) + \sum_{i' \in \eta(k) - \{i\}} m_{i'\to k}(x_k) \right) \tag{4.14}$$

where $\eta(k)$ is the set of indices of the neighbors of x_k. The second quantity involved in a message passing algorithm is that of *belief* values. Differently to messages, beliefs are computed at nodes, and they are computed mixing the messages arriving to

the node and its own information. In the min-sum algorithm this mixing for computing the belief b_i at a node x_i is computed as the sum of all messages arriving at the node [100, 7]:

$$b_i(x_i) = \Phi_i(x_i) + \sum_{k \in \eta(i)} m_{k \to i}(x_i) \tag{4.15}$$

Having defined messages and beliefs, a message passing algorithm works by firstly initializing belief values at each node; and secondly by sending messages between neighbor nodes iteratively until the updates on belief values converge, or until a maximum number of iterations has been reached. Here, there are two fundamental questions that has been researched over the past decades regarding message passing algorithms. Denote by $b_i^{(t)}(x_i)$ the computed belief value after iteration t ends, and denote the limit $\lim_{t \to \infty} b_i^{(t)}(x_i) = b_i^*(x_i)$ if convergence occurs. The first question is under what conditions the belief values $b_i^{(t)}(x_i)$ converge. The second question is given convergence, what interpretation can be given to the belief values $b_i^*(x_i)$. For the first question, it has been established that when the graph is singly connected or is a tree, the belief propagation defined in Equations 4.14 and 4.15 will converge [98, 74]. Furthermore, in these same conditions, the belief limits $b_i^*(x_i)$ can be used for finding correctly the configuration $\boldsymbol{x}^*$ that minimizes globally the energy of the potentials Φ_k and Ψ_{ik}, by computing each component as follows [75, 7]:

$$x_i^* = \arg\min_{x_i} b_i^*(x_i) \tag{4.16}$$

In general graphs with loops, the belief propagation scheme may not converge, or if convergence is achieved, the limit belief values may not be related to the optimal solution that minimizes the target energy [98, 100]. However, belief propagation has been shown to be extremely useful in computer vision and has been continuously researched and applied heuristically to graphs with loops, being know as Loopy Belief Propagation [7, 74, 68, 18]. In the algorithm for cyclic graphs, one has to choose an order of updating the messages, which in the most simplistic scenario corresponds to iterate over the array of all nodes in the graph. In the literature, the most used approach of belief propagation in computer vision is to use what is known as factor graphs [100, 7]. In general, a factor graph is composed by two kinds of nodes, *factor*

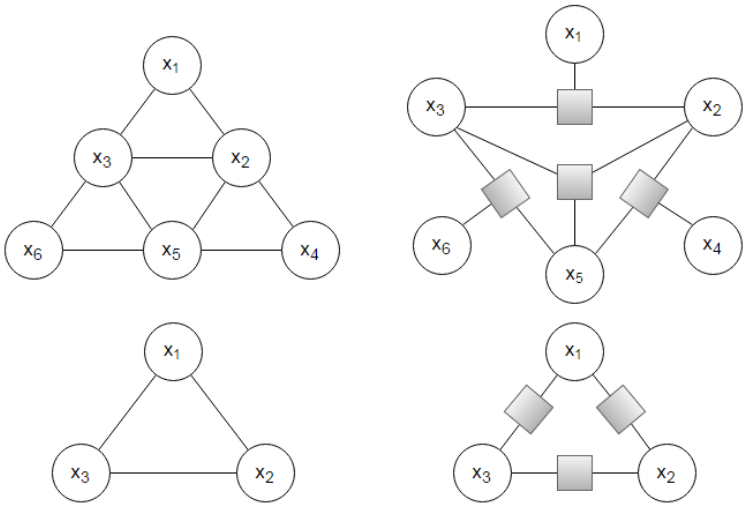


Figure 4.4: Two examples of the graph of the Vine Model in a factor graph representation. At the top, energy costs of three variables can happen at branches where a cane segment is subdivided into another two. At the bottom, an alternative is to always consider only pairwise energy factors. Boxes in this diagram represent factor nodes, which can be thought as functions that take as arguments the circular nodes that are connected to them in the graph.

nodes and *variable* nodes [100]. A variable node is simply a node in the Markov graph of the model in question. On the other hand, this Markov graph is augmented with the factor nodes which represent a factorization of the model's energy function in terms of the variables. For example, if one had an energy function of the variables x_1, x_2, x_3 of the form

$$E(x_1, x_2, x_3) = \Phi_1(x_1, x_2) + \Phi_2(x_1, x_3) + \Phi_3(x_1, x_2, x_3) \tag{4.17}$$

then there would be 3 factor nodes on the factor graph, corresponding to each cost term Φ_i; and there would be 3 variable nodes corresponding to each x_i with $i = 1, 2, 3$. An example of a factor graph for our vine Markov model is shown in Figure 4.4. As commonly used, factor graphs are represented by squares, and they are connected to the corresponding variable nodes on which the corresponding energy term is defined. Variable nodes in turn are represented by circles. For our vine model, I can have energy terms of three variables (branch points) and two variables (single connections). However, as seen in Section 3.3.6, I can also consider only pairwise energy terms and have always factor graphs of the form of the one shown at the bottom of Figure 4.4.

Using this new representation of factor graphs, it turns out that the algorithm for belief propagation can be done in sequential updates for messages that arrive from variable nodes to factors, followed by messages from factor nodes delivered to variable nodes [100]. An implementation of this approach is outlined in Algorithm 4.4. This algorithm is also the one used for evaluating inference of structure using belief propagation against the other methods presented in the previous sections. The message updates for factor nodes can be computed either from the sum of messages that arrive at the sender variable node from other factors different from the receptor [100]:

$$m_{i \to f}(x_i) = \sum_{f' \in \eta(i) - \{f\}} m_{f' \to i}(x_i) \tag{4.18}$$

On the other hand, messages from factors to variable nodes, take a similar form

Algorithm 4.4 Min-Sum Loopy Belief Propagation

procedure LBP(n_{max})
  for $f \leftarrow 1, n_f$ **do** ▷ iterate over factors
    $m_{i \to f}(x_i) \leftarrow$ INITIALIZEMSN(i, f, x_i) ▷ initialize factor messages
  end for
  for $i \leftarrow 1, n_i$ **do** ▷ iterate over variables
    $m_{f \to i}(x_i) \leftarrow$ INITIALIZEMSN(f, i, x_i) ▷ initialize variable messages
    $b_i(x_i) \leftarrow$ INITIALIZEBF(x_i) ▷ initialize beliefs for x_i
  end for
  for $n \leftarrow 1, n_{max}$ **do**
    for each $m_{i \to f}(x_i)$ **do** ▷ factor messages
      $m_{i \to f}(x_i) \leftarrow$ UPDATEMESSAGE(i, f, x_i) ▷ Equation 4.18
    end for
    for each $m_{f \to i}(x_i)$ **do** ▷ variable messages
      $m_{f \to i}(x_i) \leftarrow$ UPDATEMESSAGE(f, i, x_i) ▷ Equation 4.19
    end for
    for $i \leftarrow 1, n_{comp}$ **do** ▷ beliefs
      $b_i(x_i) \leftarrow$ UPDATEBELIEFS(x_i) ▷ Equation 4.15
    end for
  end for
  for $i \leftarrow 1, n_{comp}$ **do**
    $x_i = \arg\min_{x_i'} \; b_i(x_i')$ ▷ Equation 4.16
  end for
  return $\boldsymbol{x}$
end procedure

of the messages in Equation 4.14:

$$m_{f\to i}(x_i) = \min_{x_k \in X(f)-\{x_i\}} \left(f(x_i, x_k) + \sum_{i' \in \eta(f)-\{i\}} m_{i'\to f}(x_k) \right) \tag{4.19}$$

where I have denoted by $X(f)$ the set of variables on which the factor node f is defined and I am assuming always pairwise factors of the form $f(\cdot, \cdot)$. The general case follows straight forward from this equation. See also [100].

5

Comparative Results and Evaluation

5.1 Introduction

In this chapter I present the results and evaluation of my HMRF vine model and the MAP inference methods proposed in Chapters 3 and 4 respectively. With this purpose, firstly I start by defining the evaluation procedures that I have used for measuring precision and recall of the reconstructed vine structures. Here, there are three different properties that I evaluate, which are related to the covered region of the reconstructed canes against ground truth canes that have been manually annotated.

Having a way to measure precision and recall, secondly, I present results of applying my methods to vine images that have been extracted from vineyards at Lincoln University in Christchurch, New Zealand. Here, I compare results of the four inference methods I selected in Chapter 4. Also, I compare against my prior published research that I have generalized in this thesis.

Finally, I analyze the proposed energy model and the convergence of the energy minimization methods that I have used. For this used a test example of a vine structure that can be minimized globally in terms of the energy, and analyze whether our energy model is in accordance to the expected vine structure. I end by presenting graphs for convergence of energy changes over iterations of the inference methods.

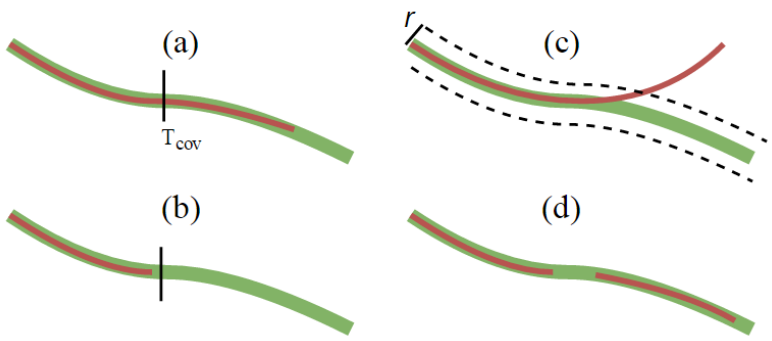


Figure 5.1: Examples of measuring precision and recall. The green lines represent ground truth canes. The red line represent inferred canes by our method. There are four different scenarios when measuring precision. In (a) more than $T_{cov}\%$ of the ground truth cane is covered by the inferred cane, while in (b) less than $T_{cov}\%$ is covered. Subsequently, in (c) more than $T_{cov}\%$ is covered but the inferred cane has $p_d < T_d\%$ points outside the ground truth cane. Finally, in (d) the ground truth cane is covered by more than one inferred cane. See the text for details.

5.2 Evaluating Vine Structure Against Ground Truth Data

In this section I present the evaluation methods to measure precision and recall of the reconstructed vine structures against ground truth data. Ground truth data consist of a set of vine images that have been manually annotated with polylines representing all canes in the images. Figure 5.2 shows an example of a ground truth vine and an inferred vine using the Gibbs Sampling method as proposed in this thesis.

To measure how accurate the inferred canes are against ground truth data, there are four scenarios that I have to account for. These are illustrated in Figure 5.1. Here, precision and recall are measured according the percentage of coverage p_{cov} of an inferred cane over a ground truth cane, and the percentage p_d of points of an inferred cane that lie at a distance of r to the points of the ground truth polylines. The basic idea of measuring precision in my methods is to threshold p_{cov} by T_{cov} and p_d by T_d according to the possible cases exposed in Figure 5.1. This thresholding is explained in the following.

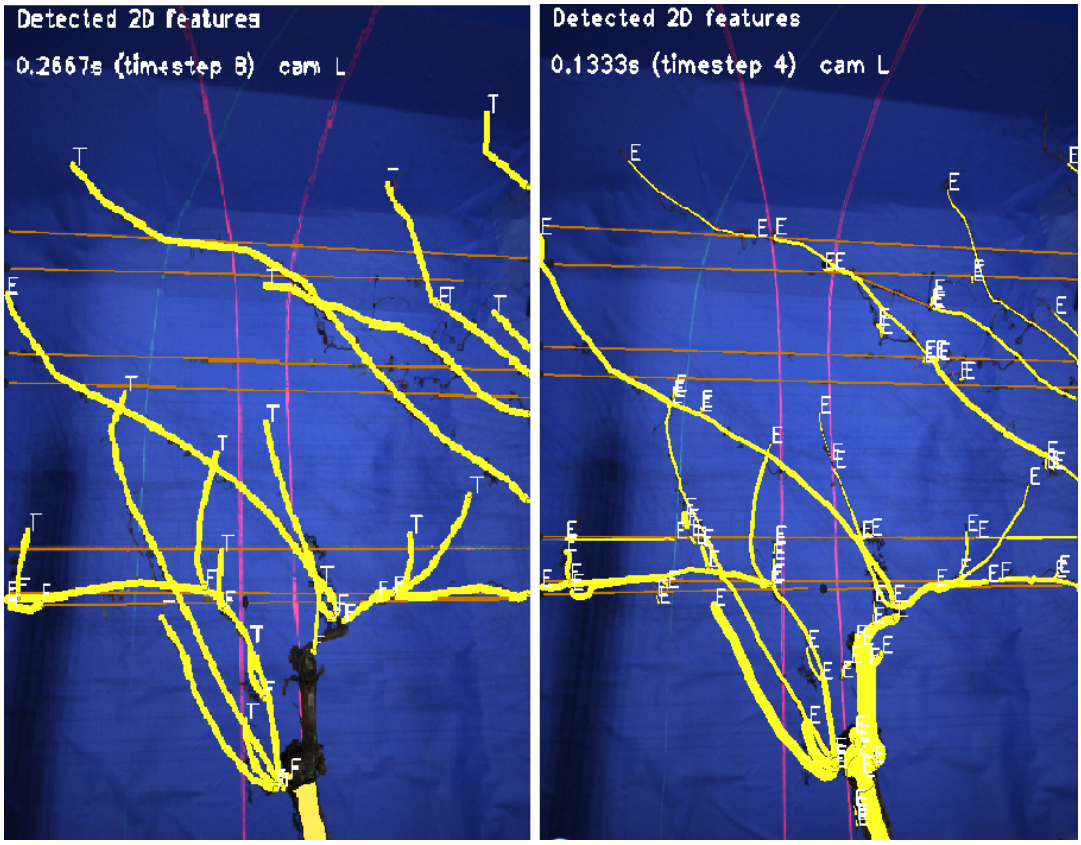


Figure 5.2: Examples of Ground Truth and Inferred Canes. To the left a ground truth cane. To the right an inferred cane using Gibbs Sampling as proposed in Section 4.3.3.

Firstly, in any of the four cases of Figure 5.1, if the percentage of coverage is $p_{cov} < T_{cov}$, then I set precision to be 0%. This means I only consider inferred canes to be correct if they cover at least $T_{cov}\%$ of a ground truth cane. For $T_{cov} = 50\%$, Figure 5.1 (a) shows a correct inferred cane, while Figure 5.1 (b) shows an incorrect inferred cane. Secondly, in the case $p_{cov} > T_{cov}$ for precision measuring, I considered as correct canes the ones that achieve $p_d > T_d$, in which case I set precision to be $p_{cov}\%$. Otherwise, if $p_d < T_d$, I set precision to 0%. Figure 5.1 (c) shows an incorrect inferred cane where $p_d < T_d$ and thus precision is 0%. Finally, in the case of Figure 5.1 (d) I simply set precision to 0% without further analysis.

The main reason for using 0% as explained above, is that I wanted to compare my results with those of [8] and my previous research [57, 58], where results used exactly this same approach. In previous research, it was found that a single ground truth cane was being represented by a very large number of small segments, which then would imply high precision even when the underlying structure of the vine was wrongly reconstructed (all segments were disconnected). Therefore, the 0% can be seen as a penalization to this case.

Similarly, another important note about our evaluation of precision and recall is that both T_d and T_{cov} are not redundant. In fact, there are scenarios where an extracted canes can go in and out of a single ground truth cane region or furthermore the extracted cane can go from one single cane into a second cane. Using only T_d this would end up being high precision, when the extracted cane is not recognizing any ground truth cane rightly. Since, we wanted to penalize this case, we used T_{cov} and we made evaluations and got results of using only T_d (by setting $T_{cov} = 0$) and using both T_d and T_{cov}. See the next section for the results.

In conclusion, for a full vine structure as a list of canes, precision is then the total length of correct inferred canes divided by the total length of inferred canes. The recall is the total length of correct inferred cane divided by the total length of ground truth canes. Here, a cane is considered correct if the percentage of coverage is more than $T_{cov}\%$ and the percentage of points inside a ground truth cane is more than T_d. In this cane the precision is set to p_{cov}. This means in Figure 5.1 the only correct cane is (a) with precision p_{cov}.

5.3 Precision-Recall Results

In this section I present precision and recall results of applying the inference methods of Chapter 4 to real vine images. For this, we had the robot collecting data at Lincoln University in Christchurch, New Zealand. We collected samples from Sauvignon Blanc type of vines available at this place. The procedure of collecting the image data from cameras is described in the introduction Chapter 1. The resulting data is in the format of videos of vines saved as image sequences. From the several thousand of image frames available from this collection, we took only 699 frames of two vine videos, for a total of around 1398 frames, though because each frame had left, right and top views, this ends with 1398 x 3 image samples. We only used this set because these were the only frames from which we have manually annotated ground truth structure of the vine. To annotate the ground truth structure for a single frame, we built a system with an user interface where a human can specify canes shown in an image and classify where there are branches, tips, crossings etc. Finally, we used this data to measure and compare precision and recall for 6 methods, including Iterated Conditional Modes (Section 4.1), Simulated Annealing (Section 4.2), the heuristic search based on Gibbs Sampling (Section 4.3.3), Belief Propagation (Section 4.4), and two prior research methods of Simulated Annealing of Marin et al. [58] and Gibbs Sampling of Marin et al. [57]. Note that by simplicity in the rest of this chapter, whenever I talk about the Gibbs Sampling method I will be referring to the heuristic search based on Gibbs Sampling presented in Section 4.3.3.

Figure 5.3 and Table 5.1 summarize all precision and recall measurements. In Figure 5.3 the preference is to have values close to the top right corner, which means high precision and high recall. I divided precision and recall measures in three categories based on the following terms:

- **Canes Overlap Coverage:** This is the percentage of points of an extracted cane that is close enough to ground truth canes, i.e., the percentage of points of an extracted cane that is overlapping ground truth canes. This is thresholded with T_d.

- **Canes Length Coverage :** This is the percentage of points of an extracted

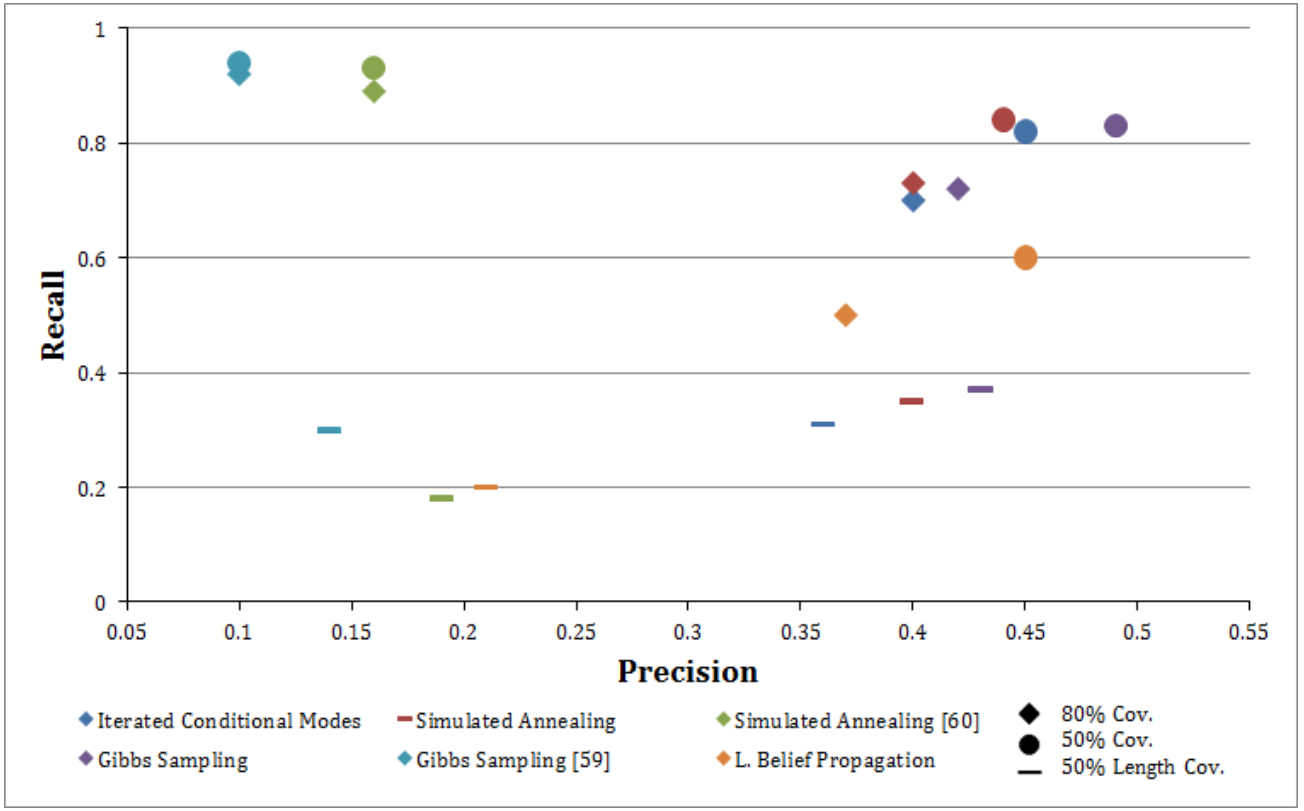


Figure 5.3: Precision and Recall results for a set of real vine images. Here a single color represent a particular method, and each type of marker (square, circle or line) represent a type of measure of the different percents of coverage of ground truth canes by inferred canes, using $T_d = 80\%$, $T_d = 50\%$ and $T_{cov} = 50\%$.

Table 5.1: Precision - Recall Results

	Precision Range min - max	**Recall Range min - max**	**Avg. Precision**	**Avg. Recall**
T_d = 50% Canes Overlap — T_{cov} = 0% Canes Length Coverage				
Iterated Conditional Modes	0.36 - 0.57	0.68 - 0.92	0.45	0.82
Simulated Annealing	0.33 - 0.64	0.75 - 0.94	0.44	0.84
Simulated Annealing [58]	0.11 - 0.32	0.63 - 0.97	0.16	0.93
Gibbs Sampling	0.39 - 0.64	0.75 - 0.95	0.49	0.83
Gibbs Sampling [57]	0.06 - 0.3	0.51- 0.95	0.1	0.94
Loopy Belief Propagation	0.32 - 0.61	0.33 - 0.85	0.45	0.6
T_d = 80% Canes Overlap — T_{cov} = 0% Canes Length Coverage				
Iterated Conditional Modes	0.31 - 0.51	0.6 - 0.89	0.4	0.70
Simulated Annealing	0.29 - 0.6	0.62 - 0.86	0.4	0.73
Simulated Annealing [58]	0.1 - 0.3	0.47 - 0.95	0.16	0.89
Gibbs Sampling	0.32 - 0.56	0.54 - 0.89	0.42	0.72
Gibbs Sampling [57]	0.03 - 0.28	0.5 - 0.95	0.1	0.92
Loopy Belief Propagation	0.24 - 0.55	0.25 - 0.89	0.37	0.5
T_d = 90% Canes Overlap — T_{cov} = 50% Canes Length Coverage				
Iterated Conditional Modes	0.16 - 0.69	0.10 - 0.66	0.36	0.31
Simulated Annealing	0.17 - 0.68	0.15 - 0.67	0.4	0.35
Simulated Annealing [58]	0.02 - 0.26	0.02 - 0.52	0.19	0.18
Gibbs Sampling	0.22 - 0.72	0.13 - 0.7	0.43	0.37
Gibbs Sampling[57]	0.01 - 0.24	0.01 - 0.55	0.14	0.30
Loopy Belief Propagation	0.05 - 0.51	0.05 - 0.48	0.21	0.2

cane that cover in length one single ground truth cane. This is thresholded with T_{cov}.

In our evaluation, we used different threshold values for T_d and T_{cov} to measure and group precision and recall in three different categories. In the first two categories, no length coverage is taken into account ($T_{cov} = 0\%$), and only canes overlapping is considered with $T_d = 50\%$, $T_d = 80\%$ respectively. On the other hand, the third category imposes not only overlapping with $T_d = 90\%$ but length coverage as well with $T_{cov} = 50\%$. Also please note that when I say that a method A have improved in more than 100% over another method B, it means that if the measure of B is x then the measure of A is at least $2 \cdot x$.

Overall, all methods presented in this thesis performed very similar in terms of canes overlap coverage with values around 0.44-0.49 for $T_d = 50\%$ and 0.37-0.4 for $T_d = 80\%$. For canes coverage in length with $T_{cov} = 50\%$, the best method is the Gibbs Sampling method with precision of 0.43 followed by Simulated Annealing with 0.4 and ICM with 0.36. Also, I observed that all the methods presented in this thesis outperformed the prior research methods of Marin et al. [57, 58]. In particular, I improved the Gibbs Sampling method [57] with increased precision by more than 100% in all three categories of coverage. Specifically increased precision from 0.1 to 0.49 when $T_d = 50\%$; from 0.1 to 0.42 when $T_d = 80\%$; and from 0.14 to 0.43 for $T_{cov} = 50\%$. Also, the Simulated Annealing method presented in this thesis, performed above the Simulated Annealing method of [58], again with improvements in 100% for all three categories. This suggest that the Simulated Annealing method in this thesis increases length coverage when including information of flow, which [58] omits.

Among the four methods presented in this thesis, belief propagation was the worst method in length coverage, in both averaged precision and recall. This suggest most of the cane segments are not being connected, and though the computed beliefs converged, they are converging far from the true MAP values (see next section). Interestingly, even this method of belief propagation performed better than prior research for recovering canes in terms of cane length coverage, with about 50% improvement against the Gibbs Sampling method [57] and 20% improvement against

the Simulated Annealing method of [58]. However, when considering only overlapping of ground truth canes, as mentioned before, belief propagation performed competitively to the rest of the methods presented here, and even surpassing all prior research methods by more than 100%.

In terms of recall, as expected, skeleton cane segments do a good job in recognizing most of the canes present in the vine images. Thus, for the cane overlapping categories, recall was above 0.7 for all methods, except for the loopy Belief Propagation method when $T_d = 80\%$ which reported recall of 0.5. Therefore, for all other methods, only 30% of the ground truth canes where not overlapped in up to $T_d = 80\%$. For the length coverage category, however, recall fell below 0.35, which suggest that even though many cane parts are recognized, several segments are not being connected.

Finally, it is important to see that overall, the vine model and all inference methods presented in this thesis surpassed previous research of Marin et al. [57, 58]. This suggest that the inclusion of penalties for maximum connections and flow contradictions that my proposed model includes helps in recognizing better vine structures in both overlapping and cane length coverage. However, all measurements of precision and recall in the third category of cane length coverage with $T_{cov} = 50\%$ fall below their respective measurements in the first two categories, since less corrected canes are detected given small unconnected cane segments fall short in representing full canes.

5.4 Experimental Analysis of the Energy Model and Convergence

In this section I want to analyze convergence of the four MAP inference methods proposed in this thesis for estimating vine structure. The outline for this experimental analysis is:

1. Get a sample set of vine binary images from which I know the true minimum energy and their structure.

2. Run each of the four MAP inference methods with random initializations and analyze energy convergence to the true minimum for all images in the sample set.

To get the sample set, unfortunately at the time of writing this thesis, there is no shortcut but to manually annotate the structure of a vine on a set of input images. After this, we would evaluate the energy on manually annotated structure states, which we would take as the true minimum energy values. As we saw in the previous section, we have already a set of manually annotated ground truth data for a considerable large number of vine images. However, the framework that was used for generating the ground truth in the previous section, is not compatible with the implementation of the methods in this thesis, as they both have different data structures and is not easy to build our Markov graph on top of them. If fact we would need the canes to be specified and cut in a way similar to the skeleton cane segments we found on Section 3.1. Another solution would be to use instead auto generated data from which we can evaluate true minimum energy. However this also implies that we need first developing believable artificial vine structures and second investigating the validity of the convergence on artificially generated data compared to true vine images. Therefore, given time restrictions in this section I performed the outlined experimental analysis on a **single vine image**. This means this section serves merely as a guideline of a work in progress and to-be-published paper on which we present the full and complete convergence analysis of all four methods. Figure 5.4 shows the single test vine image with the respective initial cane segments (all unconnected by default). We have manually annotated the true state for this image so we know exactly the true minimum energy and its structure. The image also shows the true connectivity of the cane segments which yields an energy of 30.5885. The energy model used here is the one presented in Section 3.3.6.

I start by analyzing the convergence of the method of Iterated Conditional Modes. For this, I run Algorithm 4.1 with five random initialization of both states of connections and flow directionality. As expected, since ICM is a greedy algorithm, it suffers from getting stuck at local minimum. This is shown in Figure 5.5. Also, energy values per iteration for all five initializations are shown in Table 5.2. The

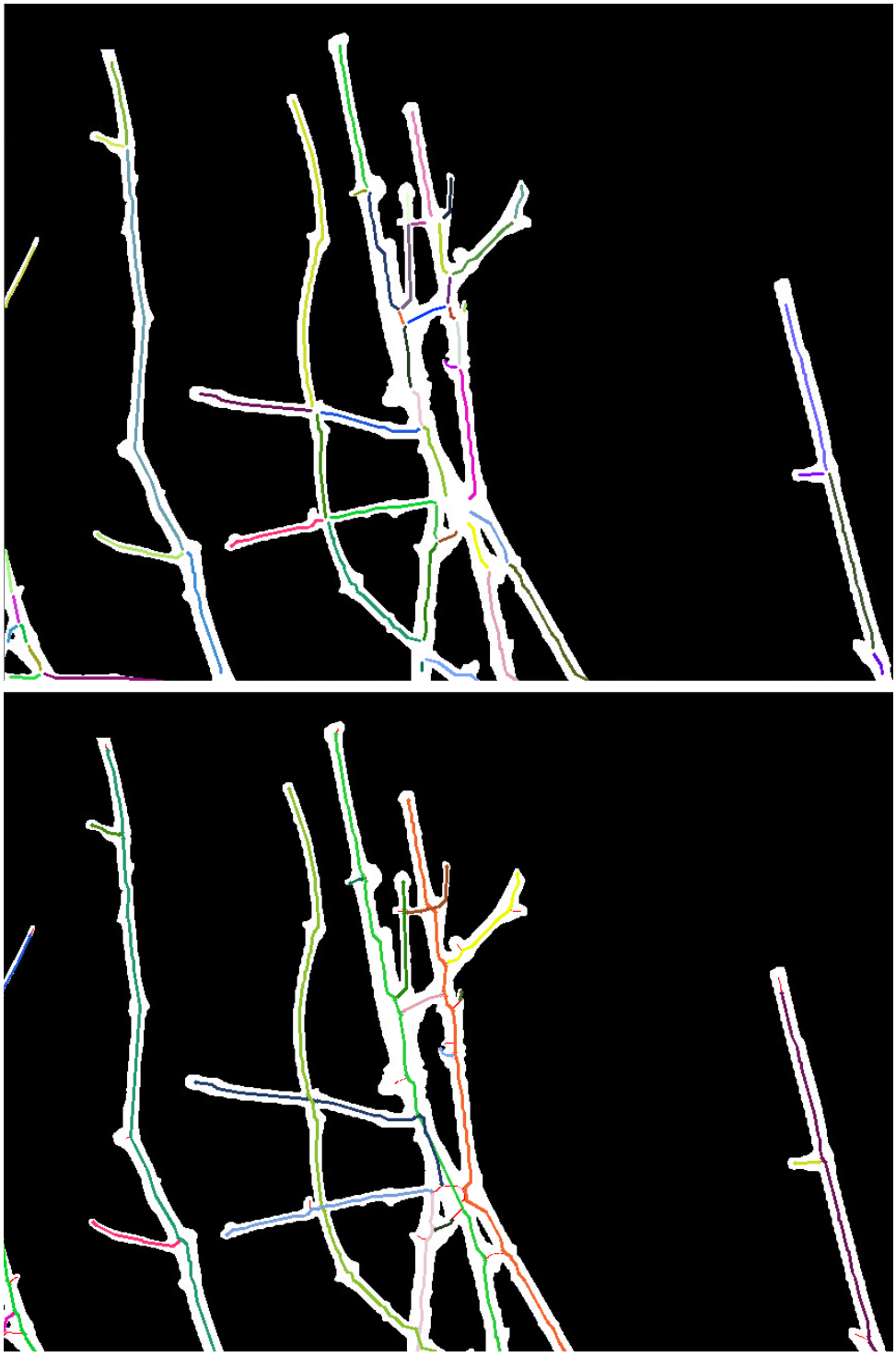


Figure 5.4: Test Vine For Energy Analysis. At the top, all skeleton cane segments that we are considering for this vine image. Here different colors represent different cane segments. At the bottom, the true vine structure with colors being full canes. Here the true minimum energy value is 30.5885.

		Initializations				
		1	**2**	**3**	**4**	**5**
Iterations	**0**	57.907	66.798	64.851	62.015	62.6995
	1	35.5462	37.1793	37.3825	39.747	40.4037
	2	35.5462	35.0217	37.3825	36.95	40.4037
	3	35.5462	35.0217	37.3825	36.95	40.4037
	4	35.5462	35.0217	37.3825	36.95	40.4037

Table 5.2: ICM with five random initializations for inferring vine structure in Figure 5.4. The algorithm achieves convergence to a local minimum in less than 5 iterations. The global minimum is 30.5885.

.

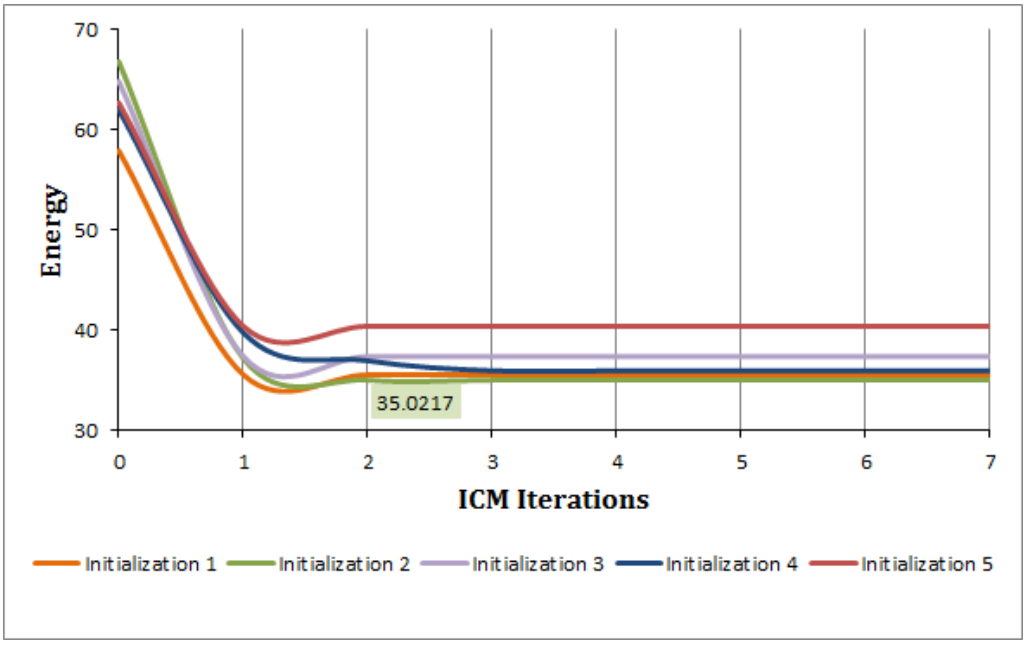


Figure 5.5: Convergence of ICM on the test image of Figure 5.4. Here each color represent a single run of the algorithm from a random initialization until convergence of the energy values.

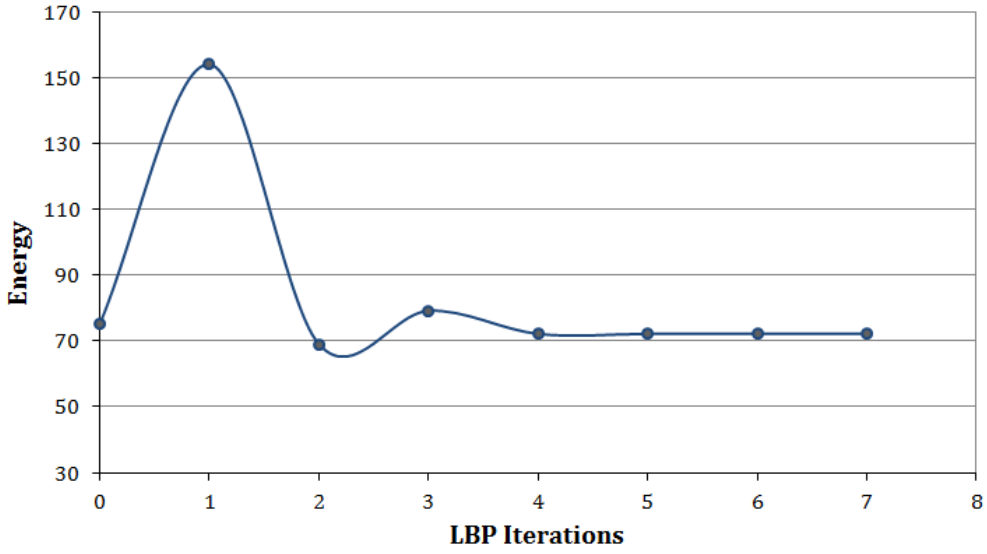


Figure 5.6: Convergence of Loopy Belief Propagation on the test image of Figure 5.4.

best achieved energy is 35.0217. I found that the algorithm always converge to a local minimum in less than 4 iterations. As the table values suggest, several different local minimum exist for our energy model. Also, ICM is very sensitive to where it is initialized, getting to different minimums even when a small difference in the initial energy. This can be seen in the energy values of initializations 4 and 5, which are 62.015 and 62.6995 and that converge to significant different energy values of 36.95 and 40.4037 respectively. One possibility for this behaviour is that the energy model yields similar energy values for vine structure configurations with significant difference in connectivity and flow directionality. Another possibility is that ICM is unstable when used for vine structure inferece, in the sense that small changes in the initial vine configuration yields big difference to the energy value to which the algorithm converges.

Similarly to ICM , loopy belief propagation presented energy convergence to a local minimum. Figure 5.6 shows the energy evolution over iterations of the algorithm. Differently from ICM , belief propagation doesn't need an initialization, and it operates taking into account all possible values for the hidden variables. The minimum energy reached by this algorithm is 68.99, which is the worst energy achieved

compared to all other three methods. Also, the energy converged quite fast, always in under 7 iterations.

Next, I analyzed the convergence of Simulated Annealing. In general, Simulated Annealing was not able to reach the global minimum on the test image of Figure 5.4, and only converged to local minimum energies around the value of 30.9 - 32.5. Given that the algorithm depends on many parameters, I ran the algorithm several times with different parameter values. The parameters to tweak are the initial temperature I_T; the minimum temperature m_T that the system is going to target during cooling; the temperature cooling rate $0 < \alpha_T < 1$ that allows the system to gradually bring the temperature down by the operation $T' = \alpha_T \cdot T$; and the maximum number of iterations that the system spends at each temperature value. In Table 5.3 and Figure 5.7 the maximum number of iterations per temperature value modified, and for each of its values, I ran simulated annealing 100 times and measured the averaged minimum energy reached. Similarly, in Table 5.4 and Figure 5.8 the cooling rate parameter α_T was modified and the averaged minimum energy was measured. For all of these experiments, all other parameters were left constants with values of minimum temperature $m_T = 0.001$, initial temperature $I_T = 2$ and $\alpha_T = 0.8$ for the first experiment and 150 maximum number of iterations on the second experiment. The results of these experiments, as seen in Figures 5.7 and 5.8 respectively, suggest two things. Firstly, the more number of iterations spent at a temperature value, the better for the system to get to lower energies. The relation seems to decay exponentially, and no big significant changes in the minimum energy is achieved after 150 - 200 maximum iterations. Secondly, as expected, the faster the systems cools down, the worst it is for it to get to lower energies and the system get stuck in local minimums. Here the relation appears to be linear and values of $\alpha_T < 0.7$ yield already solutions far from the global optimum. In conclusion, a good set of parameters for running simulated annealing for inferring vine structures is $m_T = 0.001$, $I_T = 2$, $\alpha_T = 0.8$ and 150 - 200 maximum number of iterations per temperature.

To see the iterative behaviour of thes simulated annealing algorithm over time, Figures 5.9 and 5.10 shows the evolutions for the first 12 the last 4 temperature values when running the algorithm on the test image of Figure 5.4 and with the set

# Iterations / Temperature	Avg. Min Energy
50	39.25
100	34.54
150	32.09
200	31.85
250	31.53
300	31.49

Table 5.3: Averaged Minimum Energy dependency on the maximum number of iterations per temperature in Simulated Annealing. Other parameters for the algorithm were left constants with minimum temperature $m_T = 0.001$, initial temperature $I_T = 2$ and temperature decreasing factor $\alpha_T = 0.8$.

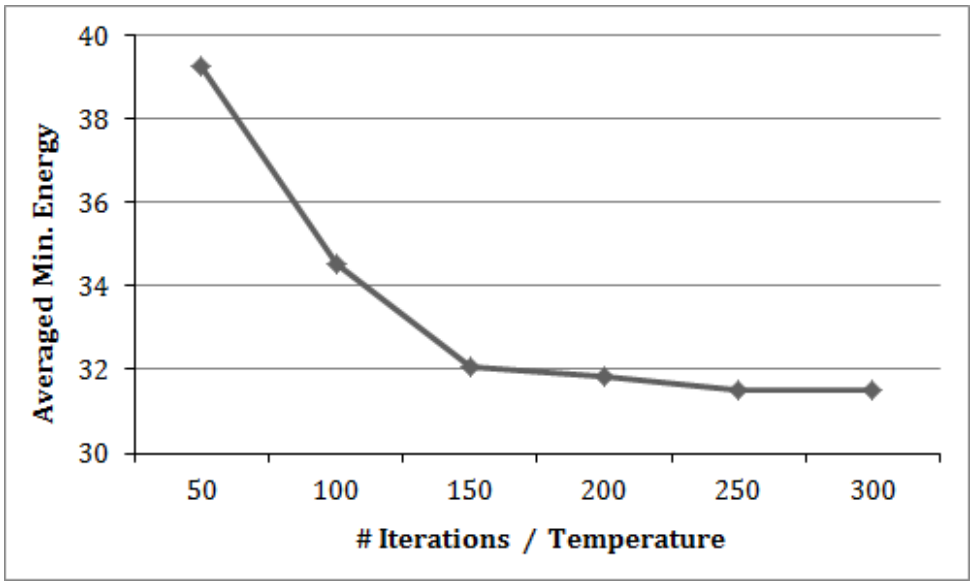


Figure 5.7: Averaged Minimum Energy reached with Simulated Annealing as a function of maximum number of iterations at each temperature value.

α_T	**Avg. Min Energy**
0.9	31.9
0.8	32.25
0.7	34.38
0.6	36.72
0.5	39.69

Table 5.4: Averaged Minimum Energy dependency on the temperature cooling rate α_T in Simulated Annealing. Other parameters for the algorithm were left constants with minimum temperature $m_T = 0.001$, initial temperature $I_T = 2$ and 150 iterations per temperature.

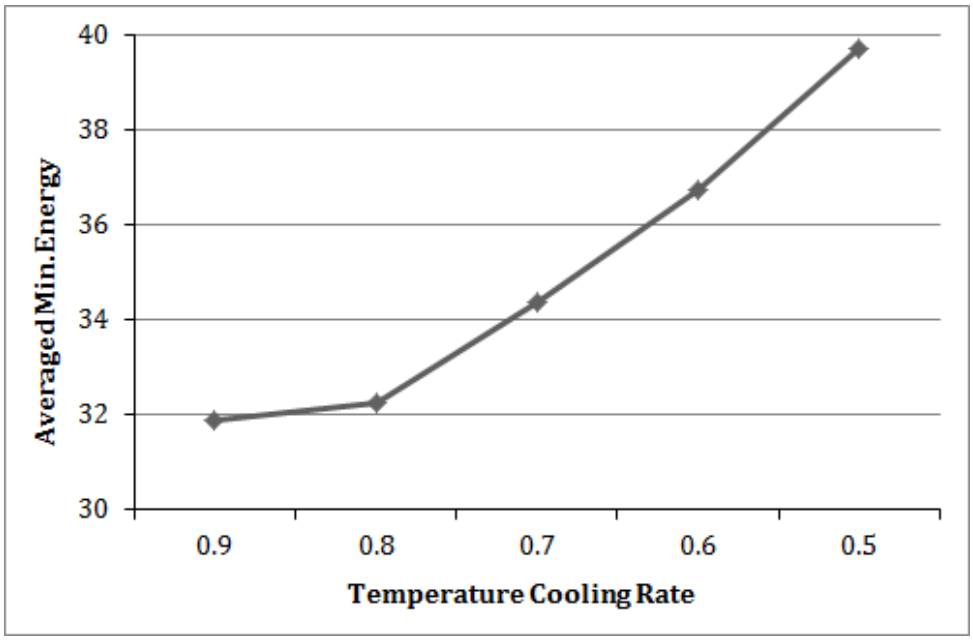


Figure 5.8: Averaged Minimum Energy reached with Simulated Annealing as a function on the temperature cooling rate α_T.

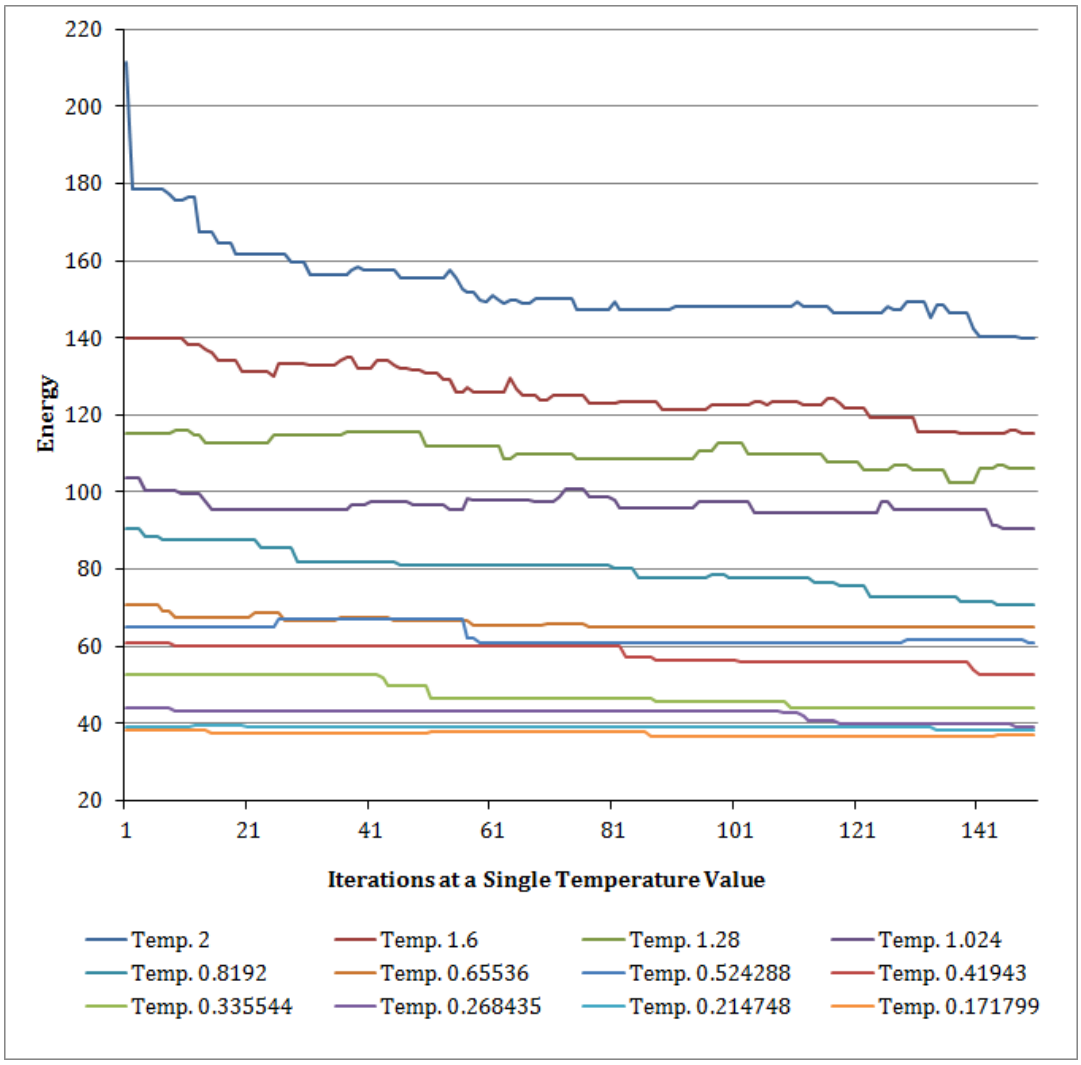


Figure 5.9: One run of the Simulated Annealing algorithm. This graph shows the cooling process, plotting the energy in each iteration of the first 12 temperature values on the vine image in Figure 5.4. Here a single color plot the energy values at a single temperature value.

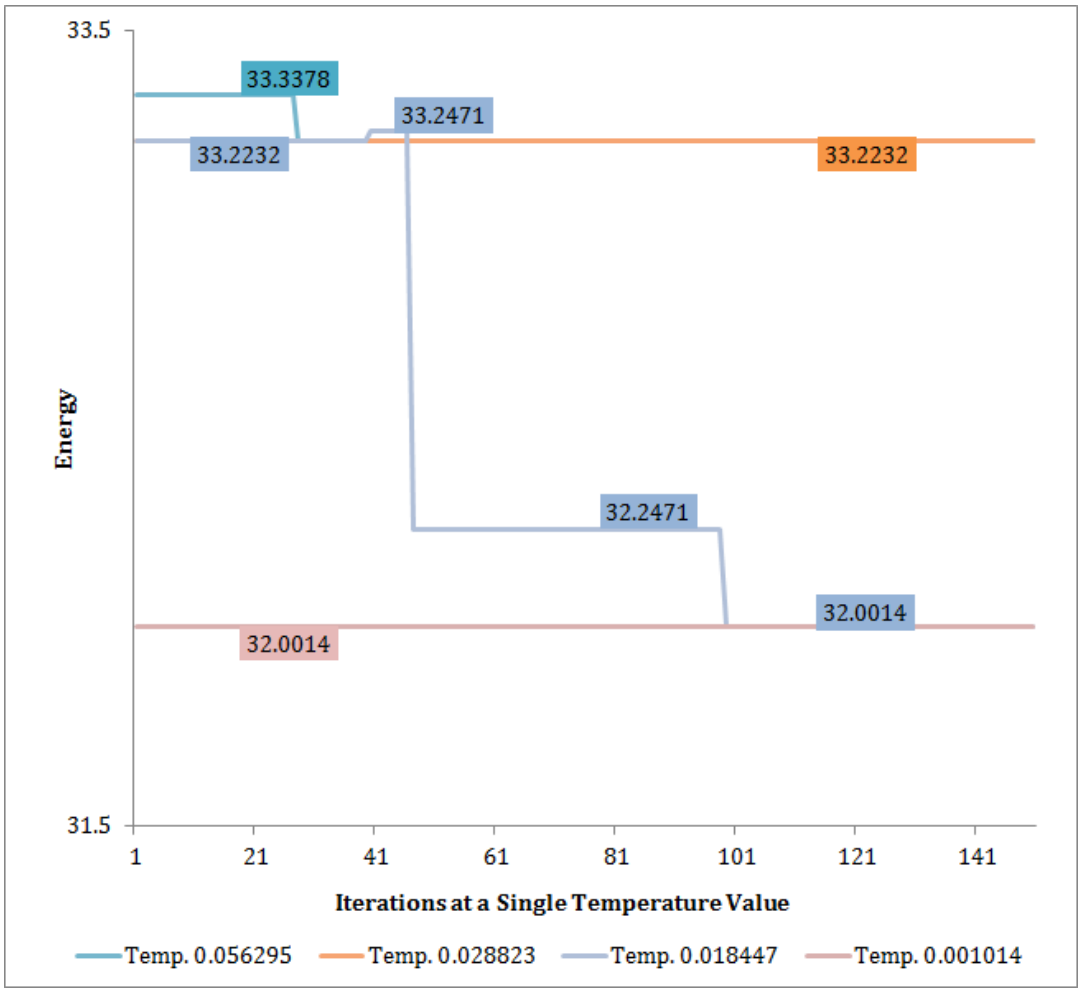


Figure 5.10: The cooling process of Simulated Annealing in the same run of the algorithm as shown in Figure 5.9. Here I show the last temperatures when convergence to a local minima is achieved. A single color plot the energy values at a single temperature value.

of parameters as described in the previous paragraph. The minimum energy reached was that of 32.0014. As shown in Figure 5.9, during high temperatures, as expected, the system oscillates down and up. Also, the system presents several horizontal regions that can arrive from two possibilities– the system is staying at a single vine structure configuration or the system is actually making changes to the structure but the changes don't affect the energy. The second option can arrive from the fact that when changing flow directionality of a whole cane, no flow contradiction is inducted, yet, the configuration is already in another state from which it can go further to other energy states.

Also, one of the main features of simulated annealing, of being able to escape local minimum to get to lower energies, can be appreciated in Figure 5.10. Here, the system is almost completely cool and its making mostly decisions to get to lower energy configurations. At a temperature of 0.056295, the system goes down from 33.3378 to 33.2232. Afterwards the system spends all the iterations of the temperature of 0.028823 at the same value, and when the new temperature of 0.0184 starts, the system escapes from this state to a higher energy state of 33.2471, only to come down to 32.2471 afterwards and finally reaching the local minimum 32.0014 where it spends all the remaining iterations.

Simulated Annealing and ICM are specifically designed to minimize an energy function. On the other hand, as I noted in Section 3.2 and Section 4.3.3, Gibbs sampling is designed to extract samples from a joint distribution, using iteratively conditional samples of one variable given all the others. The trace plot of using Gibbs Sampling for the test vine in Figure 5.4 is shown in Figure 5.11. In total the algorithm iterated for getting 20000 samples. The first thing to note in this trace plot, is that the samples generated by Gibbs sampling are presenting convergence to a stationary distribution. This can be seen at the top image, where the chain samples appear to be fluctuating and centered around the energy value of 32-33. Secondly, the chain is mixing well, in the sense that it is exploring well the distribution and getting vine structure configurations where the energy is very high. Another observation is that there is not burn in time, and the distribution of samples appear to be the same right from the first thousand iterations. Our random heuristic search, interestingly,

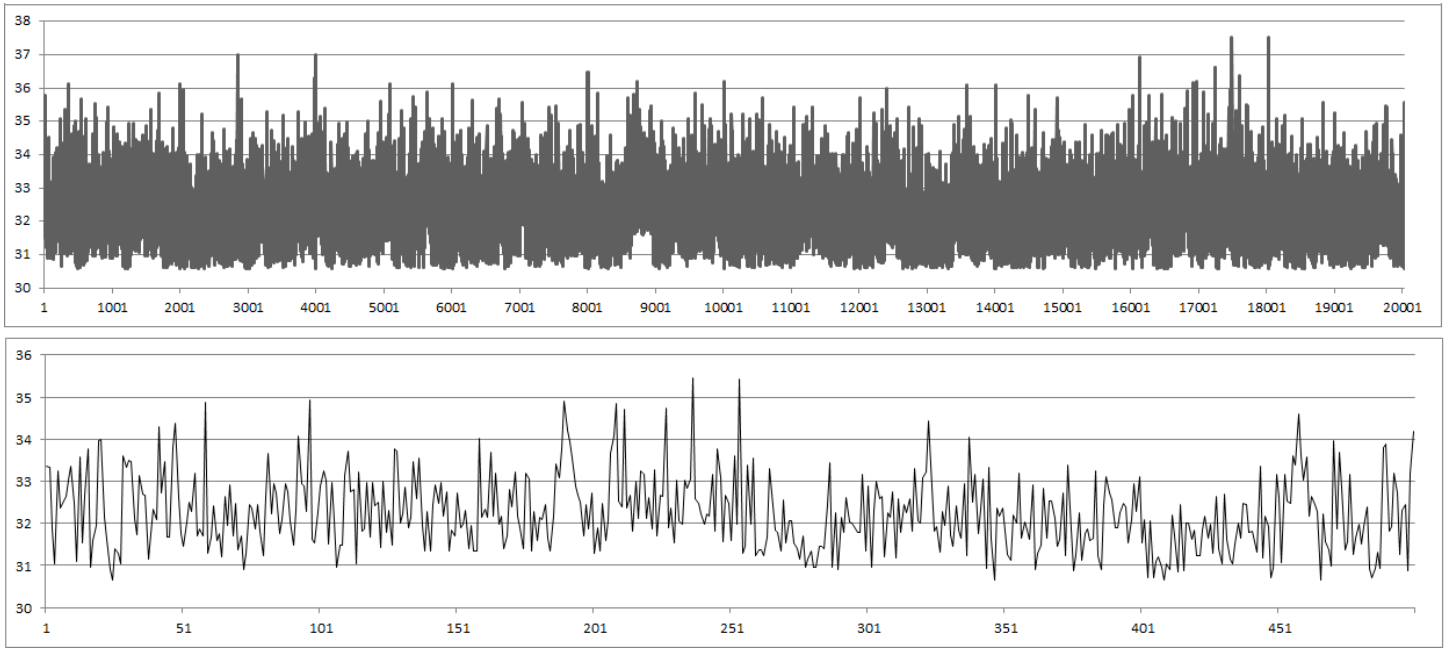


Figure 5.11: Trace Plot of Gibbs Sampling Iterations and Energy for each sample. At the top, the whole set of 20000 samples. At the bottom I zoomed in to show the last 500 samples. Right from the first iterations the minimum energy of 30.5885 for the vine structure in Figure 5.4 is sampled.

visits the global minimum energy configuration (with energy 30.5885) in the first few iterations. At a local scale, in the bottom figure, the chain samples appear to present auto correlation, since successive samples are not to far away from each other in energy terms. However, as mentioned before the chain explores quickly different energy ranges and presents convergence.

6

Conclusions

6.1 Conclusion

This thesis proposes a novel HMRF vine model to extract vine structure from images. This can be regarded as the application of state-of-the-art computer vision methods to the problem of vine structure retrieval from images. I have formulated the problem in the context of MAP inference in a way that the most likely vine structure can be inferred. The model uses skeletonisation for extracting a list of cane segments that are visible in the input images, and MAP estimation resolve hidden connectivity between them. Also, an energy model was proposed and defined by penalizing the number of connections at end points, penalizing flow contradictions between connected cane segments, and by learning two class conditional densities for modeling the probability of connecting a given pair of candidate points.

To evaluate and analyse the usefulness of our method, in this thesis I have also presented comparative results between four different MAP inference methods that solve vine structure using our proposed HMRF vine model. These methods are Iterated Conditional Modes, Simulated Annealing, an heuristic random search based on Gibbs Sampling and Belief Propagation. I have found that my proposed methods improve precision against prior research by more than 100%, meaning that if previ-

ous research had x precision, here my methods achieved at least $2 \cdot x$ in precision. Also on average, the search based on Gibbs Sampling has better averaged precision of 0.43 for length coverage in relation to the other methods, followed closely by my proposed Simulated Annealing method with averaged precision of 0.4. Also, within my methods, Belief Propagation presented the worst performance with an averaged precision of 0.21 in a scale from 0 to 1.

Note that it is hard to tell if 0.43 of precision is enough without considering the whole robot system. My system is dependent on the preprocessing steps of background removal and wires and posts extractions (to get a cleaner vine binary map). It is intuitive obvious that the better the accuracy of extracting the 2D vine structure on images, the better one would expect the stereo correspondence to work for reconstructing the 3D model of the vine using these 2D structures. However, depending on the stereo correspondence method, there could be room for fixing the inaccurate 2D structures during the reconstruction process. Therefore, the usefulness of a precision of 0.43 still needs to be addressed in the future.

In conclusion, the model and methods proposed in this thesis generalize and improve precision while maintaining consistently recall in comparison to prior research. I presented an experimental analysis of the convergence of each of the selected inference methods for the task of vine structure of a test vine image from which I have the exact minimum energy. Here, once more I assert that the heuristic random search based on Gibbs Sampling is suitable for vine structure retrieval and reaches the minimum energy value, in comparison from the other methods that usually only find a local optimum state.

6.2 Future Work

There are several topics related to my methods and results that need to be researched in the future. Firstly, we need to investigate to what extend my methods are dependent on the quality of the binary vine map. Heuristically, we noted that our algorithms worked better on images where less noise from leafs regions or artifacts are present in the binary image, but this needs to be researched in deep. One way of achieving this is to measure precision and recall on the same image with artifacts

and without artifacts, and compare.

Also, some regions are challenging to process since crossing canes with similar direction are represented in a binary image with a single skeleton cane segment, and this unique cane segment is used for representing two different real cane segments. In the future we need to address this case scenario in our model. For this, first I need to be able to recognize these regions, and then one solution could be to just duplicate the cane segment. A second solution is to modify the the Markov graph model at those regions, so a cane segment may have multiple connections not only at branching points but in these scenarios as well.

Note also that as we saw at the end of the previous Chapter, we are currently working on a paper related to the experimental analysis of convergence of the MAP inference methods. We are working to use both ground truth data and artificially generated vine structures to evaluate and compare convergence all our methods. In this work, we will be also interested in investigating and evaluating methods based on graph-cuts for estimating vine structures in our Markov model.

Finally I believe that our method can be improved in terms of precision and recall. For this I will improve the energy model to include global information of currently connected segments at any given time during optimization. Our energy model is still making mainly local decisions in the sense that a connection candidate is aware only of the first and second degree neighbors, but not on the current whole connected cane. Another limitation of the proposed method is that some cane segments are not considered even as candidates, because of the method of selecting the candidates is filtering points within a threshold distance. Also, irrelevant connection candidates are being chose like for example pairs of cane points that are separated by a big background hole. Therefore, further research for selecting connection candidates is needed.

Bibliography

[1] C. Andrieu. An introduction to MCMC for machine learning. In *14th European Conference on Machine Learning*, ECML '03, pages 5–43, Cavtat-Dubrovnik, Croatia, 2003.

[2] N. P. B. Gorte. 3d image processing to reconstruct trees from laser scans. In *Proc. Conf. of the Advanced School for Computing and Imaging*, ASCI '04, Delft, The Netherlands, 2004.

[3] D. Barthélémy and Y. Caraglio. Plant Architecture: A Dynamic, Multilevel and Comprehensive Approach to Plant Form, Structure and Ontogeny. *Annals of Botany*, 99(3):375–407, 2007.

[4] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, 48(3):48–259, 1986.

[5] E. Bienenstock, S. Geman, and D. Potter. Compositionality, MDL priors, and object recognition. In *Advances in Neural Information Processing Systems*, NIPS '96, pages 838–844, Denver, CO, USA, 1996.

[6] J. Binney and G. Sukhatme. 3D tree reconstruction from laser range data. In *IEEE International Conference on Robotics and Automation*, ICRA '09, pages 1321–1326, Kobe, Japan, 2009.

[7] A. Blake, P. Kohli, and C. Rother. *Markov Random Fields for Vision and Image Processing*. The MIT Press, Jul 2011.

[8] T. Botterill, R. Green, and S. Mills. Finding a vine?s structure by bottom-up parsing of cane edges. In *Proceedings of the 28th Conference on Image and Vision Computing New Zealand*, IVCNZ '13, pages 1–6, Wellington, New Zealand, 2013.

[9] S. Brooks and A. Gelman. *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, 1st edition, May 2011.

[10] A. Bucksch. A Practical Introduction to Skeletons for the Plant Sciences. *Applications in Plant Sciences*, 2(8):1–10, 2014.

[11] H. Bunke and A. Sanfeliu. *Syntactic and Structural Pattern Recognition: Theory and Applications*. World Scientific Pub Co Inc, Jan 1990.

[12] X. Chen, B. Neubert, Y.-Q. Xu, O. Deussen, and S. B. Kang. Sketch-based Tree Modeling Using Markov Random Field. In *Proceedings of the ACM SIGGRAPH Asia '08*, pages 1–9, New York, NY, USA, 2008.

[13] Z.-L. Cheng, X.-P. Zhang, and B.-Q. Chen. Simple Reconstruction of Tree Branches from a Single Range Image. *Journal of Computer Science and Technology*, 22(6):846–858, 2007.

[14] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[15] P. Clifford. Markov random fields in statistics. In *Disorder in Physical Systems: A Volume in Honour of John M. Hammersley*, pages 19–32. Oxford University Press, Oxford, 1990.

[16] D. Dey, L. Mummert, and R. Sukthankar. Classification of plant structures from uncalibrated image sequences. In *IEEE Workshop on Applications of Computer Vision '12*, pages 329–336, Breckenridge, CO, USA, 2012.

[17] P. Felzenszwalb. Object detection grammars. In *Proceedings of the 2011 IEEE International Conference on Computer Vision*, ICCV '11, pages 691–691, Barcelona, Spain, 2011.

[18] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006.

[19] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '08, Anchorage, Alaska, 2008.

[20] P. F. Felzenszwalb, R. B. Girshick, and D. Mcallester. Cascade object detection with deformable part models. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI '10, pages 1627–1645, United States, 2010.

[21] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[22] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *Int. J. Comput. Vision*, 61(1):55–79, 2005.

[23] M. A. Fischler and R. Elschlager. The representation and matching of pictorial structures. *Computers, IEEE Transactions on*, C-22(1):67–92, 1973.

[24] W. Gilks and S. Richardson. *Markov Chain Monte Carlo in Practice* . Chapman and Hall/CRC, Dec 1995.

[25] R. Girshick and J. Malik. Training deformable part models with decorrelated features. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, pages 3016–3023, Washington, DC, USA, 2013.

[26] R. B. Girshick. *From Rigid Templates to Grammars: Object Detection with Structured Models*. PhD thesis, University of Chicago, Chicago, IL, USA, 2012. AAI3513455.

[27] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/ rbg/latent-release5/.

[28] R. B. Girshick, P. F. Felzenszwalb, and D. A. Mcallester. Object detection with grammar models. In *Advances in Neural Information Processing Systems*, NIPS '11, pages 442–450, 2011.

[29] W. Gittoes. Vine model fitting using a composite skeletonisation method. Technical report, University of Canterbury, 2011.

[30] W. Gittoes, T. Botterill, and R. Green. Quantitative analysis of skeletonisation algorithms for agricultural modelling of branches. In *Proceedings of the 26th Conference on Image and Vision Computing New Zealand*, IVCNZ '11, pages 1–6, Parnell, Auckland, New Zealand., 2011.

[31] C. Godin and Y. Caraglio. A Multiscale Model of Plant Topological Structures. *Journal of Theoretical Biology*, 191(1):1–46, 1998.

[32] C. Godin and H. Sinoquet. FunctionalâĂŞstructural plant modelling. *New Phytologist*, 166(3):705–708, 2005.

[33] B. Gorte. Skeletonization of laser-scanned trees in the 3d raster domain. In *Lecture Notes in Geoinformation and Cartography*, 3D-GIS '06, pages 371–380, 2006.

[34] J. Guénard, G. Morin, F. Boudon, and V. Charvillat. Reconstructing Plants in 3D from a Single Image Using Analysis-by-Synthesis. In *Advances in Visual Computing*, Lecture Notes in Computer Science '13, pages 322–332, Barcelona, Spain, 2013.

[35] C. E. Guo, S. C. Zhu, and Y. N. Wu. Primal sketch: Integrating texture and structure. In *Proceedings of International Conference on Computer Vision*, ICCV '03, pages 5–19, 2003.

[36] F. Han and S.-C. Zhu. Bayesian reconstruction of 3d shapes and scenes from a single image. In *Proceedings of the First IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, HLK '03, Washington, DC, USA, 2003.

[37] F. Han and S.-C. Zhu. Bottom-up/top-down image parsing by attribute graph grammar. In *Proceedings of the 2005 IEEE International Conference on Computer Vision*, ICCV '05, pages 1778–1785, Beijing, China, 2005.

[38] F. Han and S.-C. Zhu. Bottom-up/top-down image parsing with attribute grammar. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(1):59–73, 2009.

[39] J. Hanan and P. Prusinkiewicz. Foreword: Studying plants with functional-structural models. *Functional Plant Biology*, 35(9-10):I–iii, 2008.

[40] M. Jaeger and P. H. De Reffye. Basic concepts of computer simulation of plant growth. *Journal of Biosciences*, 17(3):275–291, 1992.

[41] T. Jebara. Map estimation, message passing, and perfect graphs. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 258–267, Arlington, Virginia, United States, 2009.

[42] S. Kang. Image-Based and Sketch-Based Modeling of Plants and Trees. In *Lecture Notes in Computer Science and Computer Vision*, ACCV '10, pages 350–354, Queenstown, New Zealand, 2011.

[43] S. B. Kang and L. Quan. Image-Based Modeling of Plants and Trees. *Synthesis Lectures on Computer Vision*, 1(1):1–83, 2009.

[44] M. KoziÅĎski and R. Marlet. Image parsing with graph grammars and markov random fields applied to facade analysis. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, WACV '14, pages 729–736, 2014.

[45] P. Kumar, J. Cai, and S. Miklavcic. High-throughput 3D Modelling of Plants for Phenotypic Analysis. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*, IVCNZ '12, pages 301–306, New York, NY, USA, 2012.

[46] H. Laga and S. J. Miklavcic. Curve-based Stereo Matching for 3D Modeling of Plants. In *Proceedings of the 20th International Congress on Modelling and Simulation*, Adelaide, Australia, 2013.

[47] J. Lasserre and C. M. Bishop. Generative or discriminative? getting the best of both worlds. In *Proceedings of Bayesian Statistics '07*, pages 3–24, 2007.

[48] S. Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer Publishing Company, Incorporated, 3rd edition, Mar 2009.

[49] W. Liu, G. Kantor, F. de la Torre, and N. Zheng. Image-based tree pruning. In *IEEE International Conference on Robotics and Biomimetics*, ROBIO '12, pages 2072–2077, Guangzhou, China, 2012.

[50] X. Liu, Y. Zhao, and S.-c. Zhu. Single-view 3d scene parsing by attributed grammar. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 684–691, Washington, DC, USA, 2014.

[51] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. El-Sana. Automatic reconstruction of tree skeletal structures from point clouds. In *Proceedings of ACM SIGGRAPH Asia '10*, pages 1–8, New York, NY, USA, 2010.

[52] L. D. Lopez, D. Shantharaj, H. B. Lu Liu, and J. Yu. Robust image-based 3d modeling of root architecture. In *Proceedings of Computer Graphics International Conference*, CGI '11, Ottawa, Ontario, Canada, 2011.

[53] Y. D. Lopez, L. D. Modeling complex unfoliaged trees from a sparse set of images. In *Computer Graphics Forum*, CGF '10, pages 2075–2082, 2010.

[54] S. Lu, X. Guo, J. Du, and W. Wen. Reconstruction of tree structure from multi-scale measurement data. In *Proceedings of the World Automation Congress*, WAC '14, pages 855–860, Kona, Hawaii, 2014.

[55] N. MacDonald. *Trees and Networks in Biological Models*. Wiley-Interscience, Mar 1983.

[56] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, Oct 2003.

[57] R. Marin, T. Botterill, and R. Green. Gibbs sampling for 2d cane structure extraction from images. In *Proceedings of the 6th International Conference on Automation, Robotics and Applications*, ICARA '15, pages 461–465, Queenstown, New Zealand, 2015.

[58] R. Marin and R. Green. A hidden markov model for modeling and extracting vine structure in images. In *Proceedings of the 30th Conference on Image and Vision Computing New Zealand*, IVCNZ '15, Auckland, New Zealand, 2015.

[59] T. Matsuyama and V. Hwang. Sigma: A framework for image understanding integration of bottom-up and top-down analyses. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI '85, pages 908–915, San Francisco, CA, USA, 1985.

[60] B. Neubert, T. Franken, and O. Deussen. Approximate Image-based Tree-modeling Using Particle Flows. In *Proceedings of ACM SIGGRAPH '07*, New York, NY, USA, 2007.

[61] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems*, NIPS '02, pages 841–848, 2002.

[62] Z. Ni and T. F. Burks. Plant or tree reconstruction based on stereo vision. In *Annual meeting of the American Society of Agricultural and Biological Engineers*, ASABE '13, Kansas City, Missouri, USA, 2013.

[63] Y. Ong and W. Kurth. A graph model and grammar for multi-scale modelling using XL. In *IEEE International Conference on Bioinformatics and Biomedicine Workshops*, BIBMW '12, pages 1–8, Philadelphia, PA, USA, 2012.

[64] Y. Ong, K. Streit, M. Henke, and W. Kurth. An approach to multiscale modelling with graph grammars. *Annals of Botany*, 114(4):813–827, 2014.

[65] A. Paproki, J. Fripp, O. Salvado, X. Sirault, S. Berry, and R. Furbank. Automated 3D Segmentation and Analysis of Cotton Plants. In *International Conference of Digital Image Computing Techniques and Applications*, DICTA '11, pages 555–560, Queensland, Australia, 2011.

[66] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, Sep 2007.

[67] C. Preuksakarn, F. Boudon, P. Ferraro, J.-B. Durand, E. Nikinmaa, and C. Godin. Reconstructing Plant Architecture from 3D Laser scanner data. In *6th International Workshop on Functional-Structural Plant Models*, pages 12–17, Davis, United States, 2010.

[68] S. J. D. Prince. *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, 1st edition, Jun 2012.

[69] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag New York, Inc., Mar 1996.

[70] S. Qi. *Analysis by Synthesis: 3D Image Parsing Using Spatial Grammar and Markov Chain Monte Carlo*. PhD thesis, UCLA University, CA, USA, 2015.

[71] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang. Image-based plant modeling. *ACM Trans. Graph.*, 25(3):599–604, 2006.

[72] D. Reniers and A. Telea. Quantitative comparison of tolerance-based feature transforms. In *First Int. Conference on Computer Vision, Theory and Applications*, VISAPP '06, pages 107–114, Funchal, Madeira, Portugal, 2006.

[73] R. Rosales and S. Sclaroff. Combining generative and discriminative models in a framework for articulated pose estimation. *International Journal of Computer Vision*, 67(3):251–276, 2006.

[74] N. Ruozzi and S. Tatikonda. Convergent and correct message passing schemes for optimization problems over graphical models. In *Proceedings of the 26th Conference Annual Conference on Uncertainty in Artificial Intelligence*, UAI '10, pages 500–501, Catalina Island, CA, 2010.

[75] N. Ruozzi and S. Tatikonda. Convergent and correct message passing schemes for optimization problems over graphical models. In *Proceedings of JMLR 2010*, pages 5860–5881, 2010.

[76] T. Sakaguchi. Botanical tree structure modeling based on real image set. In *Proceedings of ACM SIGGRAPH '98 Conference abstracts and applications*, New York, NY, USA, 1998.

[77] T. Sakaguchi and J. Ohya. Modeling and animation of botanical trees for interactive virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '99, pages 139–146, New York, NY, USA, 1999.

[78] P. Salamon, R. Frost, and P. Sibani. *Facts, conjectures, and improvements for simulated annealing*. SIAM, Oct 2002.

[79] A. Schilling, S. Anja, and H.-G. Maas. Principal curves for tree topology retrieval from tls data. In *Proceedings of SILVILASER '12*, pages 16–19, 2012.

[80] A. Schilling and H.-G. Maas. Automatic reconstruction of skeletal structures from tls forest scenes. In *Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, ISPRS '14, pages 321–328, 2014.

[81] W. Shi and C. Cheung. Performance evaluation of line simplification algorithms for vector generalization. *The Cartographic Journal*, 43(1):27–44, 2006.

[82] S. E. Shimony. Finding maps for belief networks is np-hard. *Artif. Intell.*, 68(2):399–410, 1994.

[83] I. Shlyakhter, M. Rozenoer, J. Dorsey, and S. Teller. Reconstructing 3d tree models from instrumented photographs. *IEEE Comput. Graph. Appl.*, 21(3):53–61, 2001.

[84] R. Sievanen, C. Godin, T. M. DeJong, and E. Nikinmaa. FunctionalâĂŞstructural plant models: a growing paradigm for plant studies. *Annals of Botany*, 114(4):599–603, 2014.

[85] G. Stiny and J. Gips. Shape grammars and the generative specification of painting and sculpture. In *Proceedings of IFIP Congress*, Amsterdam, 1971.

[86] R. Strzodka and A. Telea. Generalized distance transforms and skeletons in graphics hardware. In *Proceedings of the Sixth Joint Eurographics - IEEE TCVG Conference on Visualization*, VISSYM '04, pages 221–230, Aire-la-Ville, Switzerland, Switzerland, 2004.

[87] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(6):1068–1080, 2008.

[88] P. Tan, T. Fang, J. Xiao, P. Zhao, and L. Quan. Single Image Tree Modeling. In *Proceedings of the SIGGRAPH Asia '08*, pages 1–7, New York, NY, USA, 2008.

[89] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan. Image-based tree modeling. *ACM Trans. Graph.*, 26(3), 2007.

[90] A. Telea and J. J. van Wijk. An augmented fast marching method for computing skeletons and centerlines. In *Proceedings of the Symposium on Data Visualisation*, VISSYM '02, page 251âĂŞ259, Aire-la-Ville, Switzerland, Switzerland, 2002.

[91] C.-H. Teng, Y.-S. Chen, and W.-H. Hsu. Tree segmentation from an image. In *Proceedings of the Conference on Machine Vision and Applications*, IAPR '05, Nagoya, Japan, 2005.

[92] W.-H. Tsai and K.-S. Fu. Attributed grammar-a tool for combining syntactic and statistical approaches to pattern recognition. *Systems, Man and Cybernetics, IEEE Transactions on*, 10(12):873–885, 1980.

[93] Z. Tu, X. Chen, A. Yuille, and S.-C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. *International Journal of Computer Vision*, 63(2):113–140, 2005.

[94] I. Ulusoy and C. Bishop. Comparison of generative and discriminative techniques for object detection and classification. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '05, pages 173–195, 2005.

[95] N. N. Vo and A. F. Bobick. From stochastic grammar to bayes network: Probabilistic parsing of complex activity. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 2641–2648, Columbus, OH, USA, 2014.

[96] J. Vos, J. B. Evers, G. H. Buck-Sorlin, B. Andrieu, M. Chelle, and P. H. B. de Visser. Functional-structural plant modelling: a new versatile tool in crop science. *Journal of experimental botany*, 61(8):2101–15, 2010.

[97] C. Wang, N. Komodakis, and N. Paragios. Markov random field modeling, inference & learning in computer vision & image understanding: A survey. *Comput. Vis. Image Underst.*, 117(11):1610–1627, 2013.

[98] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12(1):1–41, 2000.

[99] H. Xu, N. Gossett, and B. Chen. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. Graph.*, 26(4), 2007.

[100] J. Yedidia. Message-passing algorithms for inference and optimization. *Journal of Statistical Physics*, 145(4):860–890, 2011.

[101] K. C. You and K.-S. Fu. A syntactic approach to shape recognition using attributed grammars. *Systems, Man and Cybernetics, IEEE Transactions on*, 9(6):334–345, 1979.

[102] X. Zhang, H. Li, M. Dai, W. Ma, and L. Quan. Data-Driven Synthetic Modeling of Trees. *Visualization and Computer Graphics, IEEE Transactions on*, 20(9):1214–1226, 2014.

[103] Y. Zhao and S.-C. Zhu. Image parsing via stochastic scene grammar. In *Advances in Neural Information Processing Systems*, NIPS '11, Granada Spain, 2011.

[104] Y. Zhao and S.-C. Zhu. Scene parsing by integrating function, geometry and appearance models. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 3119–3126, Washington, DC, USA, 2013.

[105] Y. Zheng, S. Gu, H. Edelsbrunner, C. Tomasi, and P. Benfey. Detailed Reconstruction of 3D Plant Root Shape. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2026–2033, Washington, DC, USA, 2011.

[106] L. Zhu, C. Lin, H. Huang, Y. Chen, and A. Yuille. Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion. In *Proceedings of the European Conference in Computer Vision*, ECCV '08, pages 759–773, 2008.

[107] S.-C. Zhu and D. Mumford. A stochastic grammar of images. *Found. Trends. Comput. Graph. Vis.*, 2(4):259–362, 2006.